

F. Y. B. Sc.
(Computer Science)

Laboratory Course Work Book

Name _____

College Name _____

Roll No. _____ Division _____

Academic Year _____

Introduction

1. About the work book

This workbook is intended to be used by F.Y.B.Sc (Computer Science) students for the two Computer Science laboratory courses in their curriculum. In Computer Science, hands-on laboratory experience is critical to the understanding of theoretical concepts studied in the theory courses. This workbook provides the requisite background material as well as numerous computing problems covering all difficulty levels.

The objectives of this book are

- 1) Defining clearly the scope of the course
- 2) Bringing uniformity in the way the course is conducted across different colleges
- 3) Continuous assessment of the course
- 4) Bring in variation and variety in the experiments carried out by different students in a batch
- 5) Providing ready reference for students while working in the lab
- 6) Catering to the need of slow paced as well as fast paced learners

2. How to use this workbook

This workbook is mandatory for the completion of the laboratory course. It is a measure of the performance of the student in the laboratory for the entire duration of the course.

2.1 Instructions to the students

Please read the following instructions carefully and follow them

- 1) You are expected to carry this book every time you come to the lab for computer science practicals
- 2) A file should be maintained separately by each student which should contain the algorithms, flowcharts, written answers, source code as well as the program output.
- 3) You should prepare yourself before hand for the Exercise by reading the material mentioned



under icon

Reading



Ready Reference

- 4) If the self activity exercise or assessment work contains any blanks such as this _____, or

, get them filled by your instructor.

- 5) Instructor will specify which problems you are to solve by ticking box
- 6) Follow good programming practices like:
 - Use appropriate file naming conventions
 - Use meaningful variable names
 - Use proper Indentation
 - Use comments in the program
 - Every program should contain in comments prgrammer's name and date
- 7) You will be assessed for each exercise on a scale of 5
 - i) Not done 0
 - ii) Incomplete 1
 - iii) Late Complete 2
 - iv) Needs improvement 3
 - v) Complete 4
 - vi) Well Done 5

Instruction to the Instructors

- 1) Explain the assignment and related concepts in around ten minutes using white board if required or by demonstrating the software

- 2) Fill in the blanks with different values for each student
- 3) Choose appropriate problems to be solved by student by ticking box
- 4) Make sure that students follow the instruction as given above
- 5) After a student completes a specific set, the instructor has to verify the outputs and sign in the provided space after the activity.
- 6) Ensure that students use good programming practices.
- 7) You should evaluate each assignment carried out by a student on a scale of 5 as specified above by ticking appropriate box.
- 8) The value should also be entered on assignment completion page of the respective Lab course

Instructions to the Lab administrator

You have to ensure that appropriate hardware and software is available to each student. The operating system and software requirements on server side and also client side are as given below

- 1) Server Side (Operating System)
 - a. * Fedora Core Linux
 - * Microsoft Windows Server 2003
 - b. Servers Side (software's to be installed)
 - In Linux – C, C++, awk, shell, perl, postgresql/Mysql
 - In WinXP -- MSOffice
- 2). Client Side (Operating System)
 - a. * Red Hat Linux and Fedora Core
 - * Microsoft Windows XP
 - b. Client Side (software's to be installed)
 - In Linux – C, C++, awk, shell, perl, postgresql/mysql
 - In WinXP -- MSOffice

The detail information about installation and configuring of the server and client are provided in appendix A

3. Acknowledgements

The authors wish to express their gratitude to Dr. Narendra Jadhav, Vice Chancellor, University of Pune, for his vision and guidance in bringing out this lab book, a first of its kind. Dr. Pandit Vidyasagar, Director, Board of colleges and university department has played a pivotal role in taking this project to completion. We are indebted to Dr. V. B. Gaikwad, Dean Science Faculty, who extended his wholehearted support to this endeavor. Prof. Arun Gangarde, Chairperson, Board of studies in Computer Science deserves a special mention for his untiring efforts during the entire process.

We appreciate the efforts taken by Prof. Chitra Nagarkar , member, Board of studies in Computer Science during initial phases of the project. We would like to acknowledge the role played by the University authorities and the members of the Board of Studies in Computer Science.

Special thanks to Mr. Achyut Godbole, noted IT personality and renowned author who took a lot of interest in this project.

Our heartfelt thanks to Dr. Sanjay Kadam, CDAC and Ms. Kishori Khadilkar, Patni Computer Systems Ltd., for painstakingly reviewing the entire book and giving valuable inputs. Last but not the least, we thank all the faculty members, who have been involved in this project and shared their expertise.

Assignment Completion Sheet

Sr. No.	Title of the Assignment	Performed on	Submitted on	Remark	Marks Obtained
A) Problem Solving and C programming.					
1.	To demonstrate use of data types ,simple operators & expressions.(errors and error handling)				
2.	To demonstrate decision making statements. (if , if –else, nested if, switch control statements).				
3.	To Demonstrate loop control statements. (while, do-while, for, nested looping structures).				
4.	To demonstrate use of functions. (user defined, standard library functions & recursion).				
5.	To demonstrate use of Arrays.(1-D,2-D arrays).				
B) Database Management System.					
1.	To create simple tables using primary key constraint.				
2.	To create more than one tables with referential key Constraint, PK constraint.				
3.	To create one or more tables with General constraints.(UNIQUE, NOT NULL,CHECK).				
4.	To query tables using DROP,ALTER statements on tables.(Use of simple forms of INSERT,UPDATE,DELETE Statements).				
5.	To query the tables using the simple forms of select statements				
6.	To query the table, using SET operations. (UNION,INTERSECT).				
7.	To query tables using nested queries.(Use of EXCEPT, EXIST,NOT EXISTS,ALL clauses)				
8.	To create view.				

Practical Incharge: 1)

HoD

2)

Section I



Using basic Linux commands



You should read following topics before starting this exercise

1. UNIX and LINUX operating system
2. cat with options, ls with options, mkdir, cd, rmdir, cp, mv, cal, pwd, wc, grep with options, I/O redirection using >, >>, <, | etc.



About UNIX and LINUX

The success story of UNIX starts with the failure of the MULTICS project. The project failed and the powerful GE-645 machine was withdrawn by GE. Two scientists at Bell Labs, Ken Thompson and Dennis Ritchie, who were part of the MULTICS team, continued to work and succeeded and named their Operating system UNIX, a pun on MULTICS.

The machine available at Bell Labs was a DEC PDP-7 with only 64 k memory while the Operating system they were developing was meant for a larger machine. The problematic situation was handled with an innovative solution. They developed most part of the software in a higher level language, C, which helped them in porting their Operating system from one hardware to another.

With the growing popularity of UNIX, it was available on a variety of machines, from personal computers to mainframes. The most popular amongst them was UNIX System V from AT&T.

Each big player in the market came up with their own versions of UNIX. IBM had its own version of UNIX called AIX, which was used on high-end servers. Sun's version of UNIX called Solaris was used on Sun workstations. Novell marketed UnixWare along with Netware, its Network operating system.

LINUX is a version of UNIX, which though it resembles UNIX in looks and feels but differs from other versions in the way it was developed and distributed. In contrast to large proprietary UNIX versions, Linux was developed by Linus Torvalds, a Finnish student. He made the source code available and invited partners via the internet in his development effort. He got professional help from all quarters and Linux evolved rapidly. It was made freely available for everyone to use. Linux that was initially meant for Personal computers is now available for a variety of hardware platforms, from mainframes to handheld computers

Linux supports multiple users. Every user need to have an account in order to use the system. One of the users called system administrator (root) is given the charge of creating user accounts and managing the system normally works on the “#” prompt.

You will be given a username and password, using which you can login into Linux operating system. For computer users, the operating system provides a user-command interface that is easy to use, usually called the **Shell**. The user can type commands at the shell **prompt** and get the services of the operating system. Linux operating system shell has the “\$” prompt.

You can open a system terminal that gives you a \$ prompt where you can type in various shell commands.

LINUX system will usually offer a variety of shell types:

- sh or Bourne Shell: the original shell still used on UNIX systems and in UNIX-related environments. It is available on every Linux system for compatibility with UNIX programs.
- bash or Bourne Again shell: the standard GNU shell, is the standard shell for common users on Linux and is a *superset* of the Bourne shell.
- csh or C shell: the syntax of this shell resembles that of the C programming language.
- tcsh or Turbo C shell: a superset of the common C shell, enhancing user-friendliness and

speed.

- ksh or the Korn shell: A superset of the Bourne shell

All LINUX commands are case sensitive single words optionally having arguments. One of the argument is options which starts with “-“ sign immediately followed by one or more characters indicating option. The wild-cards or metacharacters “*” and “?” have similar meaning as in DOS. The “*” character matches any number of characters while “?” matches a single character. The backquote “ ` ” is another metacharacter. Shell executes the command enclosed in backquote in its place. Any wild-card is escaped with a \ character to be treated as it is

Shell Variables

There are number of predefined shell variables called system or environment variables which are set by the system when the system boots up. Some important system variables are

PATH	It contains set of paths where the system searches for an executable file
HOME	It is the home or login directory where the user is placed initially
PS1	It is the primary shell prompt which is usually \$
PS2	It is the secondary shell prompt which is usually >

Linux Files and directories

Linux defines three main types of files. Linux treats all devices also as files.

Ordinary or regular file	A file containing data or program
Directory file	A file containing the list of filenames and their unique identifiers
Special or device file	A file assigned to a device attached to a system

Linux files may or may not have extensions. A file can have any number of dots in its name. Linux file names are case sensitive. The root directory represented by / is the topmost directory file containing number of subdirectories which in turn contains subdirectories and files

Shell Commands

The following is the list of shell commands

Command	Used for	Example
date	Displays both date and time The command can be used by the system administrator to change date and time.	\$date Format specifiers can be used as arguments %m month in integer format %h Name of the month %d Day of the month %y Last two digits of the year %H hours %M Minutes %S Seconds \$date +%H \$date +"%h %m"
cal	Displays the calendar	\$cal 8 2007 Displays the calendar for the month august of year 2007 \$cal aug Displays the calendar for the month august of current year

cat	Displays the contents of the files used with the command	<pre>\$cat</pre> <p>Displays immediately what is typed when you hit enter key</p> <pre>\$ cat > abc.txt</pre> <p>Whatever number of lines typed till you press ^D are placed in abc.txt file</p> <pre>\$cat abc.txt</pre> <p>Displays contents of file abc.txt</p>
ls	Displays the contents of current directory. A single dot (.) stands for the current directory while a double dot(..) indicates the parent directory	<pre>\$ls</pre> <p>lists all files in the current directory</p> <pre>\$ls -a</pre> <p>Lists also the hidden files</p> <pre>\$ls -l</pre> <p>Lists the permission information along with other information such as date of last modification, size in blocks etc. The first column of the output exhibits the file type and permissions.</p> <p>File type: -, d, b respectively for ordinary, directory and block device file.</p> <p>Permissions are of the form r, w, x, - i.e. read, write, execute and none respectively.</p> <p>There are three groups of rwx. Owner, group and public.</p>
mkdir	Creates specified directory in the current directory, fails if a file or directory by that name is already present or user is not having permissions to create a directory	<pre>\$mkdir bin</pre> <p>Creates bin directory</p> <pre>\$mkdir dir1 dir2 dir3</pre> <p>Creates three directories dir1 , dir2 and dir3</p>
cd	Switches to specified directory, fails if user is not having permissions to access the directory	<pre>\$cd /</pre> <p>Switches to root directory</p> <pre>\$cd</pre> <p>Changes to HOME directory</p>
rmdir	Removes specified directory fails if the directory is not empty	<pre>\$rmdir dir1</pre> <p>Removes dir1 directory</p> <pre>\$rmdir dir2 dir3</pre> <p>Removes dir2 and dir3 directories</p>
cp	Creates an exact copy of a file with a different name	<pre>\$cp abc.txt xyz.txt</pre> <p>Copies abc.txt into a new file named xyz.txt</p> <pre>\$cp abc.txt bin</pre> <p>Copies abc.txt into a new file with the same name in bin directory</p>
mv	It renames a file or moves a group of files to a different directory	<pre>\$mv xyz.txt pqr</pre>
rm	Deletes specified file. It can be used with wildcards * and ? as in DOS, to delete all files of a specified type	<pre>\$rm pqr</pre>

pwd	Displays the path of your present working directory	\$pwd displays the directory in which you are currently working
wc	Counts words, lines and characters or bytes	\$wc -c abc.txt Displays the number of bytes in the file abc.txt \$wc -l abc.txt Displays the number of lines in the file abc.txt \$wc -w abc.txt Displays the number of words in the file abc.txt \$wc abc.txt Displays the number of bytes, words and lines in the file abc.txt
grep	The syntax is grep options pattern filename It displays the lines in the file in which the pattern is found	\$grep Agarwal names.txt Displays lines in the names.txt where the string "Agarwal" is present \$grep -n Agarwal names.txt Displays lines along with line numbers in the names.txt where the string "Agarwal" is present
man	Offers help on the shell command	\$man ls Shows entire manual page of Linux manual pertaining to ls command
passwd	It is used to change the password	\$passwd When invoked by an ordinary user asks for the old password and then demands typing and retyping of new password #passwd user1 Used by administrator to change the passwd of user1
echo	Displays its arguments compressing the spaces. To preserve the spaces the words should be placed within quotes	\$echo \$HOME \$echo \$PATH \$echo eats up the spaces \$echo The date to-day is `date` \$echo You can multiply using `*`
who	Displays list of users currently logged in	\$who
tail	Displays last lines of the file	\$tail -3 abc.txt Displays last three lines of file abc.txt
head	Displays top lines of the file	\$head -5 abc.txt Displays top five lines of file abc.txt

Redirection and pipes

The most of the above commands take some input, do some processing and give the output or give error message in case there is some error. For example the cat command is usually given as \$cat filename. Here cat command takes input from file named filename and gives output on the console. If the file is not present then it gives appropriate error message. By default the cat command writes the output or error message to the console. If we just type cat command without

any filename, it will wait for user to type characters that means, it by default is expecting input also from console. The default files where a command reads its input, sends its output and error messages are called standard input(stdin), standard output(stdout) and standard error(stderr) respectively.

By default all the above three files are attached with the terminal on which the command is executing. Therefore, every command, by default, takes its input from the keyboard and sends its output and error messages to the display screen. Redirection is used to detach default file from the command and attach some specific file. Pipes allow you to send output of one command as input to the other command. The commands that are connected via a pipe are called filters

Command	Symbol	Description	Format & Examples
Input Redirection	<	It detaches the keyboard from the standard input of command and attaches specific file	<code>\$cat < abc.txt</code> Takes its input from abc.txt and the output by default is on console. The effect is same as <code>\$cat tempfile</code>
Output Redirection	>	It detaches the console from the standard output of command and attaches specific file	<code>\$cat > file1</code> Takes its input from keyboard by default and writes the output to file1, effectively whatever typed at the keyboard goes into tempfile <code>\$cat file1 abc.txt > file2</code> The contents of file1 and abc.txt will be concatenated and send to file2 <code>\$cat file1 > /dev/lp0</code> The contents of file file1 will be sent to printer instead of console
Output Redirection without overwriting	>>	In output redirection the file is cleared before writing to it. The >> is used so that output is appended and not overwritten	<code>\$cat file1 > file1</code> The file1 contents will be cleared <code>\$cat file2 >> file2</code> The file2 will have its contents appended to it
Pipe		The pipe character is used between two commands so that output of first command is send as input to the second command	<code>\$ ls -l grep "abc"</code> Displays the line in the output of <code>ls -l</code> containing pattern abc



Execute all the commands given in the example column of all the tables above in the same order and understand the usage of the commands

Signature of the instructor

Date

 / /



Set A

1 Using cat command, create a file named 'names.txt' containing at least ten names and addresses of your friends (firstname , surname, street name, cityname). Type the following commands and explain what the command is used for and give the output of the command

Command	Explanation	Output
wc -lw names.txt		
mkdir ass1 ass2		
cp names.txt ass2		
cp names.txt list		
tail -3 list		
rmdir ass2		
cd ass2		
rm names.txt		
cd		
pwd		
ls -l		
mv list list.txt		
grep ___names.txt		

2 Using cat command create a file named college.txt containing at least ten names and location of colleges (collegename, place , pincode). Type the following commands and explain what the command is used for and give the output of the command

Command	Explanation	Output
mkdir s1 s2 s3 s4		
cp college.txt coll		
cp college.txt coll s1		
head -5 coll		
grep -n _____college.txt		
rmdir s3 s4		
cd s1		
rm coll		
pwd		
cd		
mv coll xy.txt		
rm *.txt		
ls -a		

Signature of the instructor

Date / /

Set B

Give the commands to perform the following actions and give the output

- 1 List the last three lines of the file _____
- 2 Create a file named _____containing abc.txt appended to itself
- 3 Display the current month(string) and year
- 4 Display the home directory followed by path
- 5 Write the contents of directory to a file
- 6 Append at the end of a file no of lines and the name of the file
- 7 Create a file named Manualcp containing manual for cp command

Signature of the instructor

Date / /

Set C

Give the commands to perform the following actions and verify by executing the command

- 1 Display the number of lines containing pattern "____" in first five lines of the file _____
- 2 Display the calendar of current month
- 3 Store the number of users logged-in in a file _____
- 4 Create a file containing first three and last three lines of a file.
- 5 Create a file containing word count of each and every file in the current directory plus a total at the end.
- 6 Create a single file containing the data from all .txt files in the current directory.

Signature of the instructor

Date

Exercise 1

Start Date

/ /



To demonstrate the use of data types, simple operators and expressions



You should read following topics before starting this exercise

1. Different basic data types in C and rules of declaring variables in C
2. Different operators and operator symbols in C
3. How to construct expressions in C, operator precedence
4. Problem solving steps- writing algorithms and flowcharts



1. Data type Table

Data	Data Format	C Data Type	C Variable declaration	Input Statement	Output statement
quantity month credit- card number	Numeric	int Short int long int	int quantity; short month; long ccno;	scanf("%d",&quantity); scanf("%d",&month); scanf("%ld", &ccno);	printf("The quantity is %d", quantity); printf("The credit card number is %ld, ccno);
price π	real	float double	float price; const double pi=3.141593;	scanf("%f",&price);	printf("The price is %5.2f", price);
grade	character		char grade;	scanf("%c",&grade)	printf("The grade is %c",grade);

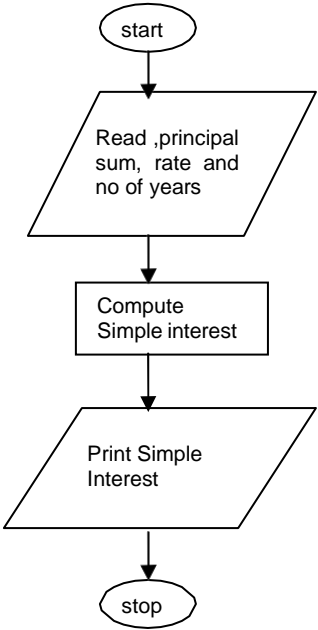
2. Expression Examples

Expression	C expression
Increment by a 3	a = a + 3
Decrement b by 1	b = b-1 or b--
$2a^2 + 5b/2$	2*a*a + 5*b/2
$7/13(x-5)$	(float)7/13*(x-5)
5% of 56	(float)5/100*56
n is between 12 to 70	n>=12 && n<=70
$\pi r^2 h$	Pi*r*r*h
n is not divisible by 7	n % 7 != 0
n is even	n%2== 0
ch is an alphabet	ch>='A' && ch<='Z' ch>='a' && ch<='z'

Note: The operators in the above expressions will be executed according to precedence and associativity rules of operators.

3. Sample program- to calculate and print simple interest after accepting principal sum, number of years and rate of interest.

Program development steps

Step 1 : Writing the Algorithm	Step 2 : Draw the flowchart	Step 3 : Writing Program
<ol style="list-style-type: none"> 1. Start 2. Accept principal sum, rate of interest and number of years 3. Compute Simple interest 4. Output Simple Interest 5. Stop 	 <pre> graph TD Start([start]) --> Read[/Read ,principal sum, rate and no of years/] Read --> Compute[Compute Simple interest] Compute --> Print[/Print Simple Interest/] Print --> Stop([stop]) </pre>	<pre> /* Program to calculate simple interest */ #include <stdio.h> main() { /* variable declarations */ float amount, rateOfInterest, simpleInterest; int noOfYears; /* prompting and accepting input */ printf("Give the Principal Sum"); scanf("%f",&amount); printf("Give the Rate of Interest"); scanf("%f",&rateOfInterest); printf("Give the Number of years"); scanf("%d",&noOfYears); /* Compute the simple Interest*/ simpleInterest=amount*noOfYears*rateOfInterest / 100; /* Print the result*/ printf("The simple Interest on amount %7.2f for %d years at the rate %4.2f is %6.2f", amount, noOfYears, rateOfInterest, simpleInterest); } </pre>



1. Type the sample program given above. Execute it for the different values as given below and fill the last column from the output given by the program. Follow the following guidelines
 - a. At \$ prompt type vi followed by filename. The filename should have .c as extension for example
\$vi pnr.c
 - b. Type the sample program given above using vi commands and save it
Compile the program using cc compiler available in Linux
\$cc pnr.c
It will give errors if any or it will give back the \$ prompt if there are no errors
A executable file a.out is created by the compiler in current directory. The program can be executed by typing name of the file as follows giving the path.
\$./a.out
Alternatively the executable file can be given name by using -o option while compiling as follows
\$cc pnr.c -o pnrexec
\$./pnrexec
The executable file by specified name will be created. Note that you have to specify the path of pnrexec as ./pnrexec , i. e., pnrexec in current (. Stands for current directory) directory otherwise it looks for program by that name in the path specified for executable programs

Sr. No	Principal sum	No of years	Rate of interest	Simple Interest
1	2000	3		
2	4500		4.5	
3	_____	6	8.3	

2. If you have not typed the program correctly, i.e., if there are syntactical errors in the program, compiler will pinpoint the errors committed and are called compile-time errors. C compiler gives line no along with error messages when it detects grammatical or syntactical errors in the program. These messages are not so straightforward and you may find it difficult to identify the error. You may miss a semicolon at the end of a statement and the compiler points out error in the next statement. You may miss just a closing '*' of a comment and it will show errors in several statements following it.
- Another type of error which is quite common is the run-time or execution error. You are able to compile the program successfully but you get run-time messages or garbage output when you execute the program.
- Modify the above program to introduce the following changes, compile, write the error messages along with line numbers, remove the error execute and indicate the type of error whether it was compile-time or execution time error.

Modified line	Error messages and line numbers	Type of error
/* Program to calculate simple interest		
int noofYears;		
scanf("%f",&amount)		
scanf("%f", amount);		
scanf("%d", noOfYears);		

Signature of the instructor

Date / /



Set A . Apply all the three program development steps for the following examples.

1. Accept dimensions of a cylinder and print the surface area and volume (Hint: surface area = $2\pi r^2 + 2\pi rh$, volume = $\pi r^2 h$)
2. Accept temperatures in Fahrenheit (F) and print it in Celsius(C) and Kelvin (K) (Hint: $C=5/9(F-32)$, $K = C + 273.15$)
3. Accept initial velocity (u), acceleration (a) and time (t). Print the final velocity (v) and the distance (s) travelled. (Hint: $v = u + at$, $s = u + at^2$)
4. Accept inner and outer radius of a ring and print the perimeter and area of the ring (Hint: perimeter = $2\pi(a+b)$, area = $\pi(a^2-b^2)$)
5. Accept two numbers and print arithmetic and harmonic mean of the two numbers (Hint: $AM=(a+b)/2$, $HM = ab/(a+b)$)
6. Accept three dimensions length (l), breadth(b) and height(h) of a cuboid and print surface area and volume (Hint : surface area= $2(lb+lh+bh)$, volume = lbh)
7. Accept a character from the keyboard and display its previous and next character in order.
Ex. If the character entered is 'd', display "The previous character is c", "The next character is e".
8. Accept a character from the user and display its ASCII value.

Signature of the instructor

Date / /

Set B . Apply all the three program development steps for the following examples.

1. Accept the x and y coordinates of two points and compute the distance between the two points.
2. Accept two integers from the user and interchange them. Display the interchanged numbers.
3. A cashier has currency notes of denomination 1, 5 and 10. Accept the amount to be withdrawn from the user and print the total number of currency notes of each denomination the cashier will have to give.

Signature of the instructor

Date

Set C. Write a program to solve the following problems

1. Consider a room having one door and two windows both of the same size. Accept dimensions of the room, door and window. Print the area to be painted (interior walls) and area to be whitewashed (roof).
2. The basic salary of an employee is decided at the time of employment, which may be different for different employees. Apart from basic, employee gets 10% of basic as house rent, 30% of basic as dearness allowance. A professional tax of 5% of basic is deducted from salary. Accept the employee id and basic salary for an employee and output the take home salary of the employee.

Signature of the instructor

Date

Assignment Evaluation

Signature

0: Not done <input type="text"/>	2: Late Complete <input type="text"/>	4: Complete <input type="text"/>
1: Incomplete <input type="text"/>	3: Needs improvement <input type="text"/>	5: Well Done <input type="text"/>

Exercise 2-a

Start Date / /



To demonstrate use of decision making statements such as if and if-else.



You should read following topics before starting this exercise

1. Different types of decision-making statements available in C.
2. Syntax for these statements.



During problem solving, we come across situations when we have to choose one of the alternative paths depending upon the result of some condition. Condition is an expression evaluating to true or false. This is known as the Branching or decision-making statement. Several forms of If and else constructs are used in C to support decision-making.

- 1) if statements
- 2) if – else
- 3) Nested if

Note: If there are more than one statement in the if or else part, they have to be enclosed in { } braces

Sr. No	Statement Syntax	Flowchart	Example
1.	<p>if statement</p> <pre>if (condition) { statement; }</pre>		<pre>if(n > 0) printf("Number is positive");</pre>
2.	<p>if - else statement</p> <pre>if (condition) { statement; } else { statement; }</pre>		<pre>if(n % 2 == 0) printf("Even"); else printf("Odd");</pre>

3.	<p>Nested if</p> <pre> if (condition) { if (condition) { statement;} else { statement;} } else { if (condition) { statement; } else { statement; } } </pre>	<pre> graph TD Start(()) --> D1{a >= b} D1 -- True --> AMax[a is max] D1 -- False --> D2{b >= c} D2 -- True --> BMax[b is max] D2 -- False --> D3{c >= a} D3 -- True --> CMax[c is max] D3 -- False --> AMax AMax --> End(()) BMax --> End CMax --> End </pre>	<pre> If (a >= b) { if (a >= c) printf(" %d is maximum",a); else printf(" %d is maximum",c); } else { if (b >= c) printf(" %d is maximum",b); else printf(" %d is maximum",c); } </pre>
----	--	---	---

4. Sample program- to check whether a number is within range.

Step 1: Writing the Algorithm	Step 2 : Draw the flowchart	Step 3 : Writing Program
<ol style="list-style-type: none"> 1. Start 2. Accept the number 3. Check if number is within range 4. if true print "Number is within range " otherwise print "number is out of range". 5. Stop 	<pre> graph TD Start((start)) --> Read[/Read number/] Read --> D{If(n in range)} D -- True --> W[/Number is within range/] D -- False --> O[/Number is out of range/] W --> Stop((stop)) O --> Stop </pre>	<pre> /* Program to check range */ #include <stdio.h> main () { /* variable declarations */ int n; int llimit=50, ulimit = 100; /* prompting and accepting input */ printf("Enter the number"); scanf("%d",&n); if(n>=llimit && n <= ulimit) printf("Number is within range"); else printf("Number is out of range"); } </pre>



1. Execute the following program for five different values and fill in the adjoining table

<pre>main() { int n; printf("Enter no."); scanf("%d",&n); if(n%__==0) printf("divisible"); else printf("not divisible"); }</pre>	n	output

2. Type the above sample program 4 and execute it for the following values.

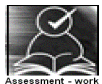
n	Output message
50	
100	
65	

3. Using the sample code 3 above write the complete program to find the maximum of three numbers and execute it for different set of values.

Instructor should fill in the blanks with appropriate values.

Signature of the instructor

Date / /



Set A: Apply all the three program development steps for the following examples.

- Write a program to accept an integer and check if it is even or odd.
- Write a program to accept three numbers and check whether the first is between the other two numbers. Ex: Input 20 10 30. Output: 20 is between 10 and 30
- Accept a character as input and check whether the character is a digit. (Check if it is in the range '0' to '9' both inclusive)
- Write a program to accept a number and check if it is divisible by 5 and 7.
- Write a program, which accepts annual basic salary of an employee and calculates and displays the Income tax as per the following rules.

Basic: < 1,50,000	Tax = 0
1,50,000 to 3,00,000	Tax = 20%
> 3,00,000	Tax = 30%
- Accept a lowercase character from the user and check whether the character is a vowel or consonant. (Hint: a,e,i,o,u are vowels)

Signature of the instructor

Date / /

Set B: Apply all the three program development steps for the following examples.

1. Write a program to check whether given character is a digit or a character in lowercase or uppercase alphabet. (Hint ASCII value of digit is between 48 to 57 and Lowercase characters have ASCII values in the range of 97 to 122, uppercase is between 65 and 90)
2. Accept the time as hour, minute and seconds and check whether the time is valid. (Hint: $0 \leq \text{hour} < 24$, $0 \leq \text{minute} < 60$, $0 \leq \text{second} < 60$)
3. Accept any year as input through the keyboard. Write a program to check whether the year is a leap year or not. (Hint leap year is divisible by 4 and not by 100 or divisible by 400)
4. Accept three sides of triangle as input, and print whether the triangle is valid or not. (Hint: The triangle is valid if the sum of each of the two sides is greater than the third side).
5. Accept the x and y coordinate of a point and find the quadrant in which the point lies.
6. Write a program to calculate the roots of a quadratic equation. Consider all possible cases.
7. Accept the cost price and selling price from the keyboard. Find out if the seller has made a profit or loss and display how much profit or loss has been made.

Signature of the instructor

Date

Set C: Write programs to solve the following problems

1. Write a program to accept marks for three subjects and find the total marks secured , average and also display the class obtained. (Class I – above __%, class II – __% to __%, pass class – __% to __% and fail otherwise)
2. Write a program to accept quantity and rate for three items, compute the total sales amount, Also compute and print the discount as follows: (amount >_ 20% discount, amount between ___to ___ -- 15% discount, amount between – ___to ___-- 8 % discount)
3. A library charges a fine for every book returned late. Accept the number of days the member is late, compute and print the fine as follows:(less than five days Rs_ fine, for 6 to 10 days Rs. ___ fine and above 10 days Rs. ___ fine)

Instructor should fill in the blanks with appropriate values.

Signature of the instructor

Date

Assignment Evaluation		Signature	
0: Not done	<input style="width: 30px; height: 20px;" type="text"/>	2: Late Complete	<input style="width: 30px; height: 20px;" type="text"/>
1: Incomplete	<input style="width: 30px; height: 20px;" type="text"/>	3: Needs improvement	<input style="width: 30px; height: 20px;" type="text"/>
		4: Complete	<input style="width: 30px; height: 20px;" type="text"/>
		5: Well Done	<input style="width: 30px; height: 20px;" type="text"/>

Exercise 2-b

Start Date / /



To demonstrate decision making statements (switch case)



You should read following topics before starting this exercise

1. Different types of decision-making statements available in C.
2. Syntax for switch case statements.



The control statement that allows us to make a decision from the number of choices is called a switch-case statement. It is a multi-way decision making statement.

1. Usage of switch statement

Statement Syntax	Flowchart	Example
<pre> switch(expression) { case value1: block1; break; case value2: block2; break; . . . default : default block; break; } </pre>	<pre> graph TD Start([start]) --> Case1{case 1} Case1 -- True --> Block1[Block 1] Case1 -- False --> Case2{case 2} Case2 -- True --> Block2[Block 2] Case2 -- False --> Case3{case 3} Case3 -- True --> Block3[Block 3] Case3 -- False --> Case4{case 4} Case4 -- True --> Block4[Block 4] Case4 -- False --> Default[Default Block] Block1 --> Stop([stop]) Block2 --> Stop Block3 --> Stop Block4 --> Stop Default --> Stop </pre>	<pre> switch (color) { case 'r' : case 'R' : printf ("RED"); break; case 'g' : case 'G' : printf ("GREEN"); break; case 'b' : case 'B' : printf ("BLUE"); break; default : printf ("INVALID COLOR"); } </pre>

2. The switch statement is used in writing menu driven programs where a menu displays several options and the user gives the choice by typing a character or number. A Sample program to display the selected option from a menu is given below.

Step 1: Writing the Algorithm	Step 2: Draw the flowchart	Step 3: Writing Program
1. Start 2. Display the menu options 3. Read choice 4. Execute statement block depending on choice 5. Stop	<pre> graph TD Start([start]) --> Display[/Display Options/] Display --> Read[/Read choice/] Read --> Case1{case 1} Case1 -- True --> S1[Statement 1] Case1 -- False --> Case2{case 2} Case2 -- True --> S2[Statement 2] Case2 -- False --> Case3{case 3} Case3 -- True --> S3[Statement 3] Case3 -- True --> Default[Default statement] Case3 -- False --> Default S1 --> Stop([stop]) S2 --> Stop S3 --> Stop Default --> Stop </pre>	<pre> /* Program using switch case and menu */ #include <stdio.h> main() { /* variable declarations */ int choice; /* Displaying the Menu */ printf("\n 1. Option 1\n"); printf(" 2. Option 2\n"); printf(" 3. Option 3\n"); printf("Enter your choice"); scanf("%d",&choice); switch(choice) { case 1: printf("Option 1 is selected"); break; case 2: printf("Option 2 is selected"); break; case 3: printf("Option 3 is selected"); break; default: printf("Invalid choice"); } } </pre>



1. Write the program that accepts a char-type variable called color and displays appropriate message using the sample code 1 above. Execute the program for various character values and fill in the following table. Modify the program to include all rainbow colours

Input character	Output Message
V	
I	
B	
G	
R	

Signature of the instructor

Date / /



Set A: Apply all the three program development steps for the following examples.

1. Accept a single digit from the user and display it in words. For example, if digit entered is 9, display Nine.
2. Write a program, which accepts two integers and an operator as a character (+ - * /), performs the corresponding operation and displays the result.
3. Accept two numbers in variables x and y from the user and perform the following operations

Options	Actions
1. Equality	Check if x is equal to y
2. Less Than	Check if x is less than y
3. Quotient and Remainder	Divide x by y and display the quotient and remainder
4. Range	Accept a number and check if it lies between x and y (both inclusive)
5. Swap	Interchange x and y

Signature of the instructor

Date / /

Set B: Apply all the three program development steps for the following examples.

1. Accept radius from the user and write a program having menu with the following options and corresponding actions

Options	Actions
1. Area of Circle	Compute area of circle and print
2. Circumference of Circle	Compute Circumference of circle and print
3. Volume of Sphere	Compute Volume of Sphere and print

2. Write a program having a menu with the following options and corresponding actions

Options	Actions
1. Area of square	Accept length ,Compute area of square and print
2. Area of Rectangle	Accept length and breadth, Compute area of rectangle and print
3. Area of triangle	Accept base and height , Compute area of triangle and print

Signature of the instructor

Date / /

Set C: Write a program to solve the following problems

1. Accept the three positive integers for date from the user (day, month and year) and check whether the date is valid or invalid. Run your program for the following dates and fill the table. (Hint: For valid date $1 \leq \text{month} \leq 12, 1 \leq \text{day} \leq \text{no-of-days}$ where no-of-days is 30 in case of months 4, 6, 9 and 11. 31 in case of months 1, 3, 5, 7, 8, 10 and 12. In case of month 2 no-of-days is 28 or 29 depending on year is leap or not)

Date	Output
12-10-1984	
32-10-1920	
10-13-1984	
29-2-1984	
29-2-2003	
29-2-1900	

2. Write a program having menu that has three options - add, subtract or multiply two fractions. The two fractions and the options are taken as input and the result is displayed as output. Each fraction is read as two integers, numerator and denominator.

Instructor should fill in the blanks with appropriate values.

Signature of the instructor

Date

Assignment Evaluation

Signature

0: Not done <input type="text"/>	2: Late Complete <input type="text"/>	4: Complete <input type="text"/>
1: Incomplete <input type="text"/>	3: Needs improvement <input type="text"/>	5: Well Done <input type="text"/>

Exercise 3-a

Start Date / /



To demonstrate use of simple loops.



You should read following topics before starting this exercise

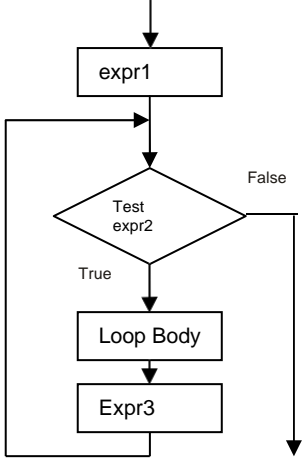
1. Different types of loop structures in C.
2. Syntax and usage of these statements.



We need to perform certain actions repeatedly for a fixed number of times or till some condition holds true. These repetitive operations are done using loop control statements. The types of loop structures supported in C are

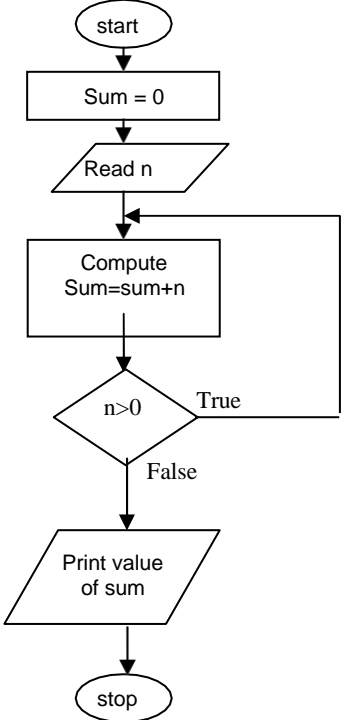
1. while statement
2. do-while statement
3. for statement

Sr. No	Statement Syntax	Flowchart	Example
1.	<p>while statement</p> <pre>while (condition) { statement 1; statement 2; . . }</pre>		<pre>/* accept a number*/ scanf("%d", &n); /* if not a single digit */ while (n > 9) { /* remove last digit n = n /10; }</pre>
2.	<p>do-while statement</p> <pre>do { statement 1; statement 2; . . } while (condition);</pre>		<pre>/*initialize sum*/ sum =0; do { /* Get a number */ printf(" give number"); scanf("%d",&n); /* add number to sum*/ sum=sum+n; } while (n>0); printf ("sum is %d", sum);</pre>

3.	for statement for(expr1; expr2; expr3) { statement 1 . . } expr1 = initialization expression expr2 = loop condition expr3 = alteration expression which alters the loop variable	 <pre> graph TD Start(()) --> Expr1[expr1] Expr1 --> Test{Test expr2} Test -- True --> LoopBody[Loop Body] LoopBody --> Expr3[Expr3] Expr3 --> Test Test -- False --> Exit(()) </pre>	/* display first 10 multiples of 2 */ for(i=1; i <= 10; i++) { printf ("2 X %d = %d\n", i, 2*i); }
----	---	--	---

Note: Usually the for loop is used when the statements have to be executed for a fixed number of times. The while loop is used when the statements have to be executed as long as some condition is true and the do-while loop is used when we want to execute statements at least once (example: menu driven programs)

3. Sample program- to print sum of 1+2+3+.....n.

Step 1: Writing the Algorithm	Step 2: Draw the flowchart	Step 3: Writing Program
1. Start 2. Initialize sum to 0. 3. Accept n. 4. Compute sum=sum+n 5. Decrement n by 1 6. if n > 0 go to step 4 7. Display value of sum. 8. Stop	 <pre> graph TD Start([start]) --> Sum0[Sum = 0] Sum0 --> ReadN[/Read n/] ReadN --> Compute[Compute Sum=sum+n] Compute --> Ngt0{n > 0} Ngt0 -- True --> Compute Ngt0 -- False --> Print[/Print value of sum/] Print --> Stop([stop]) </pre>	/* Program to calculate sum of numbers */ #include <stdio.h> main() { /* variable declarations */ int sum = 0, n; printf("enter the value of n : "); scanf("%d",&n); while (n>0) { sum = sum + n; n--; } printf("\n The sum of numbers is %d", sum); }

4. Sample program- To read characters till EOF (Ctrl+Z) and count the total number of characters entered.

Step 1 : Writing the Algorithm	Step 2 : Draw the flowchart	Step 3 : Writing Program
<ol style="list-style-type: none"> 1. Start 2. Initialize count to 0. 3. Accept ch. 4. If ch !=EOF Count = count +1 Else Go to step 6 5. Go to step 3 7. Display value of sum. 8. Stop 	<pre> graph TD Start([start]) --> Init[count = 0] Init --> Read[/Read ch/] Read --> Cond{Ch=EOF} Cond -- True --> Inc[Count = count+1] Inc --> Read Cond -- False --> Print[/Print count/] Print --> Stop([stop]) </pre>	<pre> /* Program to count number of characters */ #include <stdio.h> main() { char ch; int count=0; while((ch=getchar())!=EOF) count++; printf("Total characters = %d", count); } </pre>



1. Write a program that accepts a number and prints its first digit. Refer sample code 1 given above. Execute the program for different values.
2. Write a program that accepts numbers continuously as long as the number is positive and prints the sum of the numbers read. Refer sample code 2 given above. Execute the program for different values.
3. Write a program to accept n and display its multiplication table. Refer to sample code 3 given above.
4. Type the sample program to print sum of first n numbers and execute the program for different values of n.
5. Write a program to accept characters till the user enters EOF and count number of times 'a' is entered. Refer to sample program 5 given above.

Signature of the instructor

Date



Set A . Apply all the three program development steps for the following examples.

1. Write a program to accept an integer n and display all even numbers upto n.
2. Accept two integers x and y and calculate the sum of all integers between x and y (both inclusive)
3. Write a program to accept two integers x and n and compute x^n
4. Write a program to accept an integer and check if it is prime or not.
5. Write a program to accept an integer and count the number of digits in the number.
6. Write a program to accept an integer and reverse the number. Example: Input: 546, Output 645.
7. Write a program to accept a character, an integer n and display the next n characters.

Signature of the instructor

Date / /

Set B. Apply all the three program development steps for the following examples.

1. Write a program to display the first n Fibonacci numbers. (1 1 2 3 5)
2. Write a program to accept real number x and integer n and calculate the sum of first n terms of the series $x+ 3x+5x+7x+\dots$
3. Write a program to accept real number x and integer n and calculate the sum of first n terms of the series $\frac{1}{x} + \frac{2}{x^2} + \frac{3}{x^3} + \dots$
4. Write a program to accept characters till the user enters EOF and count number of alphabets and digits entered. Refer to sample program 5 given above.
5. Write a program, which accepts a number n and displays each digit in words. Example: 6702 Output = Six-Seven-Zero-Two. (Hint: Reverse the number and use a switch statement)

Signature of the instructor

Date / /

Set C. Write C programs to solve the following problems

1. Write a program to accept characters from the user till the user enters * and count the number of characters, words and lines entered by the user. (Hint: Use a flag to count words. Consider delimiters like \n \t , ; . and space for counting words)
2. Write a program which accepts a number and checks if the number is a palindrome (Hint number = reverse of number)
Example: number = 3472 Output: It is not a palindrome
number = 262, Output : It is a palindrome
3. A train leaves station A at 4.00 a.m and travels at 80kmph. After every 30 minutes, it reaches a station where it halts for 10 minutes. It reaches its final destination B at 1.00 p.m. Display a table showing its arrival and departure time at every intermediate station. Also calculate the total distance between A and B.

4. A task takes $4\frac{1}{2}$ hours to complete. Two workers, A and B start working on it and take turns alternately. A works for 25 minutes at a time and is paid Rs 50, B works for 75 minutes at a time and is paid Rs. 150. Display the total number of turns taken by A and B, calculate their total amounts and also the total cost of the task.

Signature of the instructor

Date / /

Assignment Evaluation

Signature

0: Not done

2: Late Complete

4: Complete

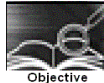
1: Incomplete

3: Needs improvement

5: Well Done

Exercise3-b

Start Date / /



To demonstrate use of nested loops



In the previous exercise, you used while, do-while and for loops. You should read following topics before starting this exercise

1. Different types of loop structures in C.
2. Syntax for these statements.
3. Usage of each loop structure



Nested loop means a loop that is contained within another loop. Nesting can be done upto any levels. However the inner loop has to be completely enclosed in the outer loop. No overlapping of loops is allowed.

Sr. No	Format	Sample Program
1.	<p>Nested for loop</p> <pre> for(exp1; exp2 ; exp3) { for(exp11; exp12 ; exp13) { } } </pre>	<pre> /* Program to display triangle of numbers*/ #include <stdio.h> void main() { int n , line_number , number; printf("How many lines: "); scanf("%d",&n); for(line_number =1 ;line_number <=n; line_number++) { for(number = 1; number <= line_number; number++) printf ("%dt", number); printf ("\n"); } } </pre>
2.	<p>Nested while loop / do while loop</p> <pre> while(condition1) { while(condition2) { } } do { while(condition1) { } } </pre>	<pre> /* Program to calculate sum of digits till sum is a single digit number */ #include <stdio.h> void main() { int n , sum; printf("Give any number "); scanf("%d",&n); do { sum =0; printf("%d --->",n); while (n>0) { sum +=n%10; </pre>

<pre>..... } while (condition2);</pre>	<pre>n= n/10; } n=sum; } while(n >9); printf (" %d" , n); }</pre>
--	--

Note: It is possible to nest any loop within another. For example, we can have a for loop inside a while or do while or a while loop inside a for.



1. The Sample program 1 displays n lines of the following triangle. Type the program and execute it for different values of n.

```

1
1 2
1 2 3
1 2 3 4
```

2. . Modify the sample program 1 to display n lines of the Floyd’s triangle as follows (here n=4).

```

1
2 3
4 5 6
7 8 9 10
```

3. The sample program 2 computes the sum of digits of a number and the process is repeated till the number reduces to a single digit number. Type the program and execute it for different values of n and give the output

Input number	Output
6534	
67	
8	
567	

Signature of the instructor

Date



Set A . Write C programs for the following problems.

1. Write a program to display all prime numbers between ___ and ___.

2. Write a program to display multiplication tables from ___ to ___ having n multiples each. The output should be displayed in a tabular format. For example, the multiplication tables of 2 to 9 having 10 multiples each is shown below.

```

2 × 1 = 2          3 × 1 = 3 .....9 × 1 = 9
2 × 2 = 4          3 × 2 = 6 .....9 × 2 = 18
.....
2 × 10 = 20       3 × 10 = 30 .....9 × 10 = 90
```

3. . Modify the sample program 1 to display n lines as follows (here n=4).

```

A      B      C      D
E      F      G
H      I
J
```

Signature of the instructor

Date / /

Set B. Write C programs for the following problems.

1. Write a program to display all Armstrong numbers between 1 and 500. (An Armstrong number is a number such that the sum of cube of digits = number itself Ex. $153 = 1*1*1 + 5*5*5 + 3*3*3$)
2. Accept characters till the user enters EOF and count the number of lines entered. Also display the length of the longest line. (Hint: A line ends when the character is `\n`)
3. Display all perfect numbers below 500. [A perfect number is a number, such that the sum of its factors is equal to the number itself]. Example: 6 (1 + 2 + 3), 28 (1+2+4+7+14)

Signature of the instructor

Date / /

Set C. Write C programs to solve the following problems

1. A company has four branches, one in each zone: North, South, East and West. For each of these branches, it collects sales information once every quarter (four months) and calculates the average sales for each zone. Write a program that accepts sales details for each quarter in the four branches and calculate the average sales of each branch.
2. A polynomial in one variable is of the form $a_0 + a_1x + a_2x^2 + \dots$. For example, $6 - 9x + 2x^5$. Write a program to calculate the value of a polynomial. Accept the number of terms n , the value of x , and $n+1$ coefficients.
3. The temperature of a city varies according to seasons. There are four seasons – spring, summer, Monsoon and winter. The temperature ranges are: Spring (15-25 degrees), Summer (25-40 degrees), Monsoon (20-35 degrees), Winter (5-20 degrees). Accept weekly temperatures (12 weeks per season) for each season, check if they are in the correct range and calculate the average temperature for each season.

Signature of the instructor

Date / /

Assignment Evaluation

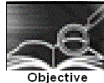
Signature

0: Not done <input type="text"/>	2: Late Complete <input type="text"/>	4: Complete <input type="text"/>
1: Incomplete <input type="text"/>	3: Needs improvement <input type="text"/>	5: Well Done <input type="text"/>

Exercise 4-a

Start Date

/ /



To demonstrate menu driven programs and use of standard library functions



You should read following topics before starting this exercise

1. Use of switch statement to create menus as in exercise 3
2. Use of while and do while loops as in exercise 4



A function is a named sub-module of a program, which performs a specific, well-defined task. It can accept information from the calling function in the form of arguments and return only 1 value. C provides a rich library of standard functions. We will explore some such function libraries and use these functions in our programs.

ctype.h : contains function prototypes for performing various operations on characters. Some commonly used functions are listed below.

Function Name	Purpose	Example
isalpha	Check whether a character is a alphabet	if (isalpha(ch))
isalnum	Check whether a character is alphanumeric	if (isalnum(ch))
isdigit	Check whether a character is a digit	if (isdigit(ch))
isspace	Check whether a character is a space	if (isspace(ch))
ispunct	Check whether a character is a punctuation symbol	if (ispunct(ch))
isupper	Check whether a character is uppercase alphabet	if (isupper(ch))
islower	Check whether a character is lowercase alphabet	if (islower(ch))
toupper	Converts a character to uppercase	ch = toupper(ch)
tolower	Converts a character to lowercase	ch = tolower(ch)

math.h : This contains function prototypes for performing various mathematical operations on numeric data. Some commonly used functions are listed below.

Function Name	Purpose	Example
cos	cosine	$a^2 + b^2 - 2*a*b*\cos(\text{abangle})$
exp(double x)	exponential function, computes e^x	exp(x)
log	natural logarithm	c= log(x)
log10	base-10 logarithm	y=log10(x)
pow(x,y)	compute a value taken to an exponent, x^y	y = 3*pow(x , 10)
sin	sine	z= sin(x) / x
sqrt	square root	delta=sqrt(b*b - 4*a*c)

Note: If you want to use any of the above functions you must include the library for example

```
#include <ctype.h>
```

```
#include <math.h>
```

In case of math library , it needs to be linked to your program. You have to compile the program as follows

```
$ cc filename -lm
```


A program that does multiple tasks, provides a menu from which user can choose the appropriate task to be performed. The menu should appear again when the task is completed so that the user can choose another task. This process continues till the user decides to quit. A menu driven program can be written using a combination of do-while loop containing a switch statement. One of the options provided in a menu driven program is to exit the program.

Statement Syntax	Flowchart	Example
<pre>do { display menu; accept choice; switch(choice) { case value1: block1; break; case value2: block2; break; . . . default : default block; } }while(choice != exit);</pre>	<pre> graph TD Start([start]) --> Display[Display menu] Display --> Accept[/Accept choice/] Accept --> Case1{case 1} Case1 -- True --> Block1[block 1] Case1 -- False --> Case2{case 2} Case2 -- True --> Block2[block 2] Case2 -- False --> Default[default block] Block1 --> Exit{choice=exit} Block2 --> Exit Default --> Exit Exit -- True --> Stop([stop]) Exit -- False --> Display </pre>	<pre>ch = getchar(); do { printf("\ n 1 : ISUPPER ") ; printf("\ n 2 : ISLOW ER ") ; printf("\ n 3 : ISDIGIT "); printf("\ n 4: EXIT"); printf("Enter your choice :"); scanf("%d", &choice); switch (choice) { case 1: if(isupper(ch)) printf(" Uppercase"); break; case 2: if(islower(ch)) printf(" Lowercase"); break; case 3: if(isdigit(ch)) printf(" Digit"); break; } }while (choice!=4);</pre>



Self-Activity

1. Write a menu driven program to perform the following operations on a character type variable.
 - i. Check if it is an alphabet
 - ii. Check if it is a digit.
 - iii. Check if it is lowercase.
 - iv. Check if it is uppercase.
 - v. Convert it to uppercase.
 - vi. Convert it to lowercase.

Refer to the sample code given above and use standard functions from ctype.h



Assessment - work

Set A . Write C programs for the following problems

1. Write a program, which accepts a character from the user and checks if it is an alphabet, digit or punctuation symbol. If it is an alphabet, check if it is uppercase or lowercase and then change the case.

2. Write a menu driven program to perform the following operations till the user selects Exit. Accept appropriate data for each option. Use standard library functions from math.h
 i. Sine ii. Cosine iii. log iv. e^x v. Square Root vi. Exit

3. Accept two complex numbers from the user (real part, imaginary part). Write a menu driven program to perform the following operations till the user selects Exit.
 i. ADD ii. SUBTRACT iii. MULTIPLY iv. EXIT

Signature of the instructor

Date / /

Set B . Write C programs for the following problems

1. Accept x and y coordinates of two points and write a menu driven program to perform the following operations till the user selects Exit.
 i. Distance between points.
 ii. Slope of line between the points.
 iii. Check whether they lie in the same quadrant.
 iv. EXIT
 (Hint: Use formula $m = (y_2 - y_1) / (x_2 - x_1)$ to calculate slope of line.)

2. Write a simple menu driven program for a shop, which sells the following items: The user selects items using a menu. For every item selected, ask the quantity. If the quantity of any item is more than 10, give a discount of _____%. When the user selects Exit, display the total amount.

Item	Price

Instructor should fill in the blanks with appropriate values.

Signature of the instructor

Date / /

Set C . Write C programs for the following problems

1. Write a program to calculate the total price for a picnic lunch that a user is purchasing for her group of friends. She is first asked to enter a budget for the lunch. She has the option of buying apples, cake, and bread. Set the price per kg of apples, price per cake, and price per loaf of bread in constant variables. Use a menu to ask the user what item and how much of each item she would like to purchase. Keep calculating the total of the items purchased. After purchase of an item, display the remaining amount. Exit the menu if the total has exceeded the budget. In addition, provide an option that allows the user to exit the purchasing loop at any time.

Signature of the instructor

Date / /

Assignment Evaluation

Signature

0: Not done 2: Late Complete 4: Complete
 1: Incomplete 3: Needs improvement 5: Well Done

Exercise 4-b

Start Date

/ /



To demonstrate writing C programs in modular way (use of user defined functions)



You should read following topics before starting this exercise

1. Declaring and Defining a function
2. Function call
3. Passing parameters to a function
4. Function returning a value



You have already used standard library functions. C allows to write and use user defined functions. Every program has a function named main. In main you can call some functions which in turn can call other functions.

The following table gives the syntax required to write and use functions

Sr. No	Actions involving functions	Syntax	Example
1.	Function declaration	returntype function(type arg1, type arg2 ...);	void display(); int sum(int x, int y);
2.	Function definition	returntype function(type arg1, type arg2 ...) { /* statements*/ }	float calcarea (float r) { float area = Pi *r*r ; return area; }
3.	Function call	function(arguments); variable = function(arguments);	display(); ans = calcarea(radius);

1. Sample code

The program given below calculates the area of a circle using a function and uses this function to calculate the area of a cylinder using another function.

```
/* Program to calculate area of circle and cylinder using function */
```

```
#include <stdio.h>
void main()
{
    float areacircle (float r);
    float areacylinder(float r, int h);
    float area, r;
    printf("\n Enter Radius: ");
    scanf("%f",&r);

    area=areacircle(r);
    printf("\n Area of circle =%6.2f", area);

    printf("\n Enter Height: ");
```

```

scanf("%d",&h);
area=areacylinder(r,h);
printf("\n Area of cylinder =%6.2f", area);
}

float areacircle (float r)
{
    const float pi=3.142;
    return(pi * r*r );
}
float areacylinder (float r, int h)
{
    return 2*areacircle(r)*h;
}

```

2. Sample code

The function `isspace` returns 1 if its character parameter is a space, tab or newline character. The program accepts characters till the user enters EOF and counts the number of white spaces.

```

/* Program to count whitespaces using function */

#include <stdio.h>
void main()
{
    int isspace (char ch);
    char ch;
    int count=0;

    printf("\n Enter the characters. Type CTRL +Z to terminate: ");
    while((ch=getchar())!=EOF)
        if(isspace(ch))
            count++;
    printf("\n The total number of white spaces =%d", count);
}
int isspace (char ch)
{
    switch(ch)
    {
        case ' ':
        case '\t':
        case '\n': return 1;
        default : return 0;
    }
}

```



1. Type the program given in sample code 1 above and execute the program. Comment function declarations and note down the type of error and the error messages received. Add another function to calculate the volume of sphere and display it.

2. Type the program given in sample code 2 above and execute the program. Comment function declaration and note down the type of error and the error messages received. Modify the function such that it returns 1 if the character is a vowel. Also count the total number of vowels entered.



Set A . Write C programs for the following problems

1. Write a function isEven, which accepts an integer as parameter and returns 1 if the number is even, and 0 otherwise. Use this function in main to accept n numbers and ckeck if they are even or odd.
2. Write a function, which accepts a character and integer n as parameter and displays the next n characters.
3. Write a function, which accepts a character and integer n as parameter and displays the next n characters.

Signature of the instructor

Date / /

Set B . Write C programs for the following problems

1. Write a function isPrime, which accepts an integer as parameter and returns 1 if the number is prime and 0 otherwise. Use this function in main to display the first 10 prime numbers.
2. Write a function that accepts a character as parameter and returns 1 if it is an alphabet, 2 if it is a digit and 3 if it is a special symbol. In main, accept characters till the user enters EOF and use the function to count the total number of alphabets, digits and special symbols entered.
3. Write a function power, which calculates x^y . Write another function, which calculates n! Using for loop. Use these functions to calculate the sum of first n terms of the Taylor series:

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} + \dots$$

Signature of the instructor

Date / /

Set C . Write C programs for the following problems

1. Write a menu driven program to perform the following operations using the Taylor series. Accept suitable data for each option. Write separate functions for each option.

- i. e^x
$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots \quad \text{for all } x$$
- ii. $\sin(x)$
$$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n + 1)!} x^{2n+1} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots \quad \text{for all } x$$
- iii. $\cos(x)$
$$\cos x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n)!} x^{2n} = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \dots \quad \text{for all } x$$

Define separate functions to calculate x^y and n! and use them in each function.

Signature of the instructor

Date / /

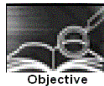
Assignment Evaluation

Signature

0: Not done <input type="checkbox"/>	2: Late Complete <input type="checkbox"/>	4: Complete <input type="checkbox"/>
1: Incomplete <input type="checkbox"/>	3: Needs improvement <input type="checkbox"/>	5: Well Done <input type="checkbox"/>

Exercise 4-c

Start Date / /



To demonstrate Recursion.



You should read the following topics before starting this exercise

1. Recursive definition
2. Declaring and defining a function
3. How to call a function
4. How to pass parameters to a function



Recursion is a process by which a function calls itself either directly or indirectly. The points to be remembered while writing recursive functions

- i. Each time the function is called recursively it must be closer to the solution.
- ii. There must be some terminating condition, which will stop recursion.
- iii. Usually the function contains an if –else branching statement where one branch makes recursive call while other branch has non-recursive terminating condition

Expressions having recursive definitions can be easily converted into recursive functions

Sr. No	Recursive definition	Recursive Function	Sample program
1.	<p>The recursive definition for factorial is given below:</p> $n! = 1 \quad \text{if } n = 0 \text{ or } 1$ $= n * (n-1)! \text{ if } n > 1$	<pre>long int factorial (int n) { If (n==0) (n==1) /* terminating condition */ return(1); else return(n* factorial(n-1)); /* recursive call */ }</pre>	<pre>#include <stdio.h> main() { int num; /* function declaration */ long int factorial(int n); printf("\n enter the number:"); scanf("%d",&num); printf("\n The factorial of %d is %ld",num,factorial(num)); } /* function code*/</pre>
2.	<p>The recursive definition for nCr (no of combinations of r objects out of n objects) is as follows</p> $nCn = 1$ $nC0 = 1$ $nCr = n-1Cr + nCr-1$	<pre>long int nCr(int n, int r) { if(n==r r==0) /* terminating condition */ return(1); else return (nCr(n-1,r) + nCr(n, r-1)); /* recursive call */ }</pre>	<pre>#include <stdio.h> /* function code*/ main() { int n, r; printf("\n enter the total number of objects:"); scanf("%d",&n); printf("\n enter the number of objects to be selected"); scanf("%d",&r); printf("\n The value %dC%d is %ld",n, r, nCr(n,r)); }</pre>



1. Write the sample program 1 given above and execute the program. Modify the program to define a global integer variable count and increment it in factorial function. Add a printf statement in main function for variable count. Execute the program for different values and fill in the following table.

Sr. No.	num	factorial	Count
1.	0		
2	1		
3	5		
4	___		
5	___		

2. Write the sample program 2 given above and execute the program for different values of n and r. Modify the program to define a global integer variable count and increment it in nCr function. Add a print statement in main function for variable count. Execute the program for different values and fill in the following table

Sr. No.	n	r	nCr	Count
1.	5	0		
2	5	5		
3	5	2		
4	5			
5	___	___		

Instructor should fill in the blanks with appropriate values.

Signature of the instructor

Date / /



Set A . Write C programs for the following problems

1. Write a recursive C function to calculate the sum of digits of a number. Use this function in main to accept a number and print sum of its digits.

2. Write a recursive C function to calculate the GCD of two numbers. Use this function in main. The GCD is calculated as :

$$\begin{aligned} \text{gcd}(a,b) &= a && \text{if } b = 0 \\ &= \text{gcd}(b, a \text{ mod } b) && \text{otherwise} \end{aligned}$$

3. Write a recursive function for the following recursive definition. Use this function in main to display the first 10 numbers of the following series

$$\begin{aligned} a_n &= 3 && \text{if } n = 1 \text{ or } 2 \\ &= 2 * a_{n-1} + 3 * a_{n-2} && \text{if } n > 2 \end{aligned}$$

4. Write a recursive C function to calculate x^y . (Do not use standard library function)

Signature of the instructor

Date / /

Set B . Write C programs for the following problems

1. Write a recursive function to calculate the nth Fibonacci number. Use this function in main to display the first n Fibonacci numbers. The recursive definition of nth Fibonacci number is as follows:

$$\begin{aligned} \text{fib}(n) &= 1 && \text{if } n = 1 \text{ or } 2 \\ &= \text{fib}(n-2) + \text{fib}(n-1) && \text{if } n > 2 \end{aligned}$$

2. Write a recursive function to calculate the sum of digits of a number till you get a single digit number. Example: 961 -> 16 -> 5. (Note: Do not use a loop)

3. Write a recursive C function to print the digits of a number in reverse order. Use this function in main to accept a number and print the digits in reverse order separated by tab.

Example 3456

6 5 4 3

(Hint: Recursiveprint(n) = print n if n is single digit number
 = print n % 10 + tab + Recursiveprint(n/10)

Signature of the instructor

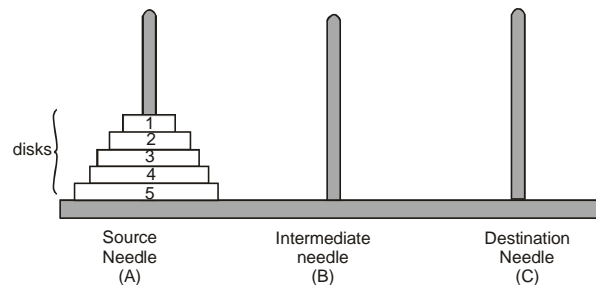
Date / /

Set C . Write C programs for the following problems

1. The "Towers of Hanoi" problem: The objective is to move a set of disks arranged in increasing sizes from top to bottom from the source pole to a destination pole such that they are in the same order as before using only one intermediate pole subject to the condition that

- Only one disk can be moved at a time
- A bigger disk cannot be placed on a smaller disk.

Write a recursive function which displays all the steps to move n disks from A to C.



Signature of the instructor

Date / /

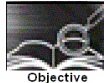
Assignment Evaluation

Signature

0: Not done <input type="text"/>	2: Late Complete <input type="text"/>	4: Complete <input type="text"/>
1: Incomplete <input type="text"/>	3: Needs improvement <input type="text"/>	5: Well Done <input type="text"/>

Exercise 5-a

Start Date / /



To demonstrate use of 1-D arrays and functions.



You should read the following topics before starting this exercise

1. What are arrays and how to declare an array?
2. How to enter data in to array and access the elements of an array.
3. How to initialize an array and how to check the bounds of an array?
4. How to pass an array to a function



An array is a collection of data items of the same data type referred to by a common name. Each element of the array is accessed by an index or subscript. Hence, it is also called a subscripted variable.

Actions involving arrays	syntax	Example
Declaration of array	<i>data-type array_name[size];</i>	int temperature[10]; float pressure[20];
Initialization of array	<i>data-type array_name[]={element1, element2,, element n};</i> <i>data-type array_name[size]={element-1, element-2,, element-size};</i> You cannot give more number of initial values than the array size. If you specify less values, the remaining will be initialized to 0.	int marks[]={45,57,87,20,90}; marks[3] refers to the fourth element which equals 20 int count[3]={4,2,9}; count[2] is the last element 9 while 4 is count[0]
Accessing elements of an array	The array index begins from 0 (zero) To access an array element, we need to refer to it as array_name[index].	Value = marks[3]; This refers to the 4 th element in the array
Entering data into an array.		for (i=0; i<=9; i++) scanf("%d", &marks[i]);
Printing the data from an array		for(i=0; i<=9; i++) printf("%d", marks[i]);
Arrays and function	We can pass an array to a function using two methods. 1. Pass the array element by element 2. Pass the entire array to the function	<i>/* Passing the whole array*/</i> void modify(int a[5]) { int i; for(i=0; i<5 ; i++) a[i] = i; }

Sample program to find the largest element of an array

```
/* Program to find largest number from array */  
  
#include<stdio.h>  
int main()  
{  
    int arr[20]; int n;  
    void accept(int a[20], int n);  
    void display(int a[20], int n);  
    int maximum(int a[20], int n);  
  
    printf("How many numbers :");  
    scanf("%d", &n);  
    accept(arr,n);  
    display(arr,n);  
    printf("The maximum is :%d" , maximum(arr,n));  
}  
  
void accept(int a[20], int n)  
{  
    int i;  
    for(i=0; i<n ; i++)  
        scanf("%d", &a[i]);  
}  
  
void display(int a[20], int n)  
{  
    int i;  
    for(i=0; i<n ; i++)  
        printf("%d\t", a[i]);  
}  
int maximum(int a[20], int n)  
{  
    int i, max = a[0];  
  
    for(i=1; i<n ; i++)  
        if(a[i] > max)  
            max = a[i];  
    return max;  
}
```



Self-Activity

1. Write a program to accept n numbers in an array and display the largest and smallest number. Using these values, calculate the range of elements in the array. Refer to the sample code given above and make appropriate modifications.

2. Write a program to accept n numbers in an array and calculate the average. Refer to the sample code given above and make appropriate modifications.

Signature of the instructor

Date



Set A. Write programs to solve the following problems

1. Write a program to accept n numbers in the range of 1 to 25 and count the frequency of occurrence of each number.
2. Write a function for Linear Search, which accepts an array of n elements and a key as parameters and returns the position of key in the array and -1 if the key is not found. Accept n numbers from the user, store them in an array. Accept the key to be searched and search it using this function. Display appropriate messages.
3. Write a function, which accepts an integer array and an integer as parameters and counts the occurrences of the number in the array.
4. Write a program to accept n numbers and store all prime numbers in an array called prime. Display this array.

Signature of the instructor

Date

Set B. Write programs to solve the following problems

1. Write a program to accept n numbers from the user and store them in an array such that the elements are in the sorted order. Display the array. Write separate functions to accept and display the array. (Hint: Insert every number in its correct position in the array)
2. Write a function to sort an array of n integers using Bubble sort method. Accept n numbers from the user, store them in an array and sort them using this function. Display the sorted array.
3. Write a program to accept a decimal number and convert it to binary, octal and hexadecimal. Write separate functions.
4. Write a program to find the union and intersection of the two sets of integers (store it in two arrays).
5. Write a program to remove all duplicate elements from an array.

Signature of the instructor

Date

Set C. Write programs to solve the following problems

1. Write a program to merge two sorted arrays into a third array such that the third array is also in the sorted order.

a1	10	25	90						
a2	9	16	22	26	10				
					0				
a3	9	10	16	22	25	26	90	100	

2. Write a program to accept characters from the user till the user enters EOF and calculate the frequency count of every alphabet. Display the alphabets and their count.

Input: THIS IS A SAMPLE INPUT

Output: Character	Count
T	2
H	1
I	3
.....	

3. Write a recursive function for Binary Search, which accepts an array of n elements and a key as parameters and returns the position of key in the array and -1 if the key is not found. Accept n numbers from the user, store them in an array and sort the array. Accept the key to be searched and search it using this function. Display appropriate messages

Signature of the instructor

Date

Assignment Evaluation

Signature

0: Not done <input style="width: 30px; height: 20px;" type="text"/>	2: Late Complete <input style="width: 30px; height: 20px;" type="text"/>	4: Complete <input style="width: 30px; height: 20px;" type="text"/>
1: Incomplete <input style="width: 30px; height: 20px;" type="text"/>	3: Needs improvement <input style="width: 30px; height: 20px;" type="text"/>	5: Well Done <input style="width: 30px; height: 20px;" type="text"/>

Exercise 5-b

Start Date / /



To demonstrate use of 2-D arrays and functions.



You should read the following topics before starting this exercise

1. How to declare and initialize two-dimensional array
2. Accessing elements
3. Usage of two dimensional arrays



Actions involving 2-D arrays	syntax	Example
Declaration of 2-D array	<i>data-type array_name[size][size];</i>	int mat[10][10]; float sales[4][10];
Initialization of 2-D array	<i>data-type array_name[rows][cols]={ {elements of row 0},{ elements of row 1},.....}; data-type array_name[][cols]={element1, element2,, element size};</i>	int num[][2] = {12, 34, 23, 45, 56, 45}; int num[3][2] = { {1,2}, {3,4}, {5,6}}; int num[3][2] = { 1,2,3,4, 5,6};
Accessing elements of 2-D array	Accessing elements of an two-dimensional array - in general, the array element is referred as <i>array_name[index1][index2]</i> where <i>index1</i> is the row location of and <i>index2</i> is the column location of an element in the array.	int m[3][2]; m is declared as a two dimensional array (matrix) having 3 rows (numbered 0 to 2) and 2 columns (numbered 0 to 1). The first element is m[0][0] and the last is m[2][1]. value = m[1][1];
Entering data into a 2-D array.		int mat[4][3]; for (i=0; i<4; i++) /* outer loop for rows */ for (j=0;j<3; j++) /* inner loop for columns */ scanf("%d", &mat[i][j]);
Printing the data from a 2-D array		for (i=0; i<4; i++) /* outer loop for rows */ { for (j=0;j<3; j++) /* inner loop for columns */ printf("%d\t", mat[i][j]); printf("\n"); }

Sample program to accept, display and print the sum of elements of each row of a matrix.

```
/* Program to calculate sum of rows of a matrix*/

#include<stdio.h>
int main()
{
    int mat[10][10], m, n;
    void display(int a[10][10], int m, int n);
    void accept(int a[10][10], int m, int n);
    void sumofrows(int a[10][10], int m, int n);

    printf("How many rows and columns? ");
    scanf("%d%d",&m, &n);

    printf("Enter the matrix elements :");
    accept(mat, m, n);
    printf("\n The matrix is :\n");
    display(mat, m, n);
    sumofrows(mat,m,n);
}

void accept(int a[10][10], int m, int n)
{
    int i,j;
    for (i=0; i<m; i++) /* outer loop for rows */
        for (j=0;j<n; j++) /* inner loop for columns */
            scanf("%d", &a[i][j]);
}

void display(int a[10][10], int m, int n)
{
    int i,j;
    printf("\nThe elements of %d by %d matrix are\n", m, n);
    for (i=0; i<m; i++) /* outer loop for rows */
    {
        for (j=0;j<n; j++) /* inner loop for columns */
            printf("%d\t", a[i][j]);
        printf("\n");
    }
}

void somofrows(int a[10][10], int m, int n)
{
    int i,j, sum;
    for (i=0; i<m; i++) /* outer loop for rows */
    { sum=0
        for (j=0;j<n; j++) /* inner loop for columns */
            sum= sum+a[i][j];
        printf("Sum of elements of row %d = %d", i, sum);
    }
}
```



1. Write a program to accept, display and print the sum of elements of each row and sum of elements of each column of a matrix. Refer to sample code given above.

Signature of the instructor

Date / /



Set A . Write C programs for the following problems.

1. Write a program to accept a matrix A of size mXn and store its transpose in matrix B. Display matrix B. Write separate functions.
2. Write a program to add and multiply two matrices. Write separate functions to accept, display, add and multiply the matrices. Perform necessary checks before adding and multiplying the matrices.

Signature of the instructor

Date / /

Set B . Write C programs for the following problems.

1. Write a menu driven program to perform the following operations on a square matrix. Write separate functions for each option.
 - i) Check if the matrix is symmetric.
 - ii) Display the trace of the matrix (sum of diagonal elements).
 - iii) Check if the matrix is an upper triangular matrix.
2. Write a menu driven program to perform the following operations on a square matrix. Write separate functions for each option.
 - i) Check if the matrix is a lower triangular matrix.
 - ii) Check if it is an identity matrix.

3. Write a program to accept an mXn matrix and display an m+1 X n+1 matrix such that the m+1th row contains the sum of all elements of corresponding row and the n+1th column contains the sum of elements of the corresponding column.

Example:

A				B			
1	2	3		1	2	3	6
4	5	6		4	5	6	15
7	8	9		7	8	9	24
				12	15	18	45

Signature of the instructor

Date / /

Set C. Write programs to solve the following problems

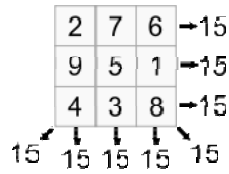
1. Pascal's triangle is a geometric arrangement of the binomial coefficients in a triangle. It is named after Blaise Pascal. Write a program to display n lines of the triangle.

```

      1
     1 1
    1 2 1
   1 3 3 1
  1 4 6 4 1
 1 5 10 10 5 1
1 6 15 20 15 6 1

```

2. A magic square of order n is an arrangement of n^2 numbers, in a square, such that the n numbers in all rows, all columns, and both diagonals sum to the same constant. A normal magic square contains the integers from 1 to n^2 . The magic constant of a magic square depends on n and is $M(n) = (n^3+n)/2$. For $n=3,4,5$, the constants are 15, 34, 65 resp. Write a program to generate and display a magic square of order n .



Signature of the instructor

Date

Assignment Evaluation

Signature

0: Not done <input style="width: 40px; height: 20px;" type="text"/>	2: Late Complete <input style="width: 40px; height: 20px;" type="text"/>	4: Complete <input style="width: 40px; height: 20px;" type="text"/>
1: Incomplete <input style="width: 40px; height: 20px;" type="text"/>	3: Needs improvement <input style="width: 40px; height: 20px;" type="text"/>	5: Well Done <input style="width: 40px; height: 20px;" type="text"/>

Section II

Exercise 1

Start Date / /



To create simple tables , with only the primary key constraint (as a table level constraint & as a field level constraint) (include all data types)



You should read following topics before starting this exercise

1. Designing relations into tables
2. Using DDL statements to create tables



A table is a database object that holds data. A table must have unique name, via which it can be referred. A table is made up of columns. Each column in the table must be given a unique name within that table. Each column will also have size a data-type and an optional constraint.

The data types permitted are

Data type	Syntax	Description	Example
Character data types	char(n)	It is fixed length character string of size n, default value 1 byte if n is not specified.	account_type char(6)
	varchar(n)	It is variable length character string with maximum size n.	employee_name varchar(50)
	Text	It is used to store large text data, no need to define a maximum	work_experience text
Numeric data types	Integer , int , serial	Serial is same as int, only that values are incremented automatically	Eno int Eno serial
	Numeric	A real number with P digits, S of them after decimal point.	Sal numeric(5,2) Sal numeric(n)
	Float	Real number	Weight Float
Date and time type	Date	Stores date information	Birthdate date
	Time	Stores time information	Birthtime time
	Timestamp	Stores a date & time	Birth timestamp
Boolean and Binary type	Boolean, bool	Stores only 2 values : true or false, 1 or 0, yes or no, y or n, t or f	Flag Boolean

Constraints can be defined as either of the following :

Name	Description	Example
Column level	When data constraint is defined only with respect to one column & hence defined after the column definition, it	Create tablename (attribute1 datatype primary key , attribute2 datatype constraint constraint-name

	is called as column level constraint	,.....)
Table Level	When data constraint spans across multiple columns & hence defined after defining all the table columns when creating or altering a table structure, it is called as table level constraint	Create tablename (attribute1 datatype , attribute2 datatype2 ,....., constraint pkey primary key(attribute1,attribute2))

Syntax for **table creation** :

Create tablename (attribute list);

Attribute list : ([attribute name data type optional constraint] ,.....)

Primary key concept :

Description	Properties	Example
A primary key is made up of one or more columns in a table, that uniquely identify each row in the table.	A column defined as a primary key, must conform to the following properties : a) The column cannot have NULL values. b) The data held across the column MUST be UNIQUE.	Create tablename (attribute1 datatype primary key , attribute2 datatype ,.....) Create tablename (attribute1 datatype , attribute2 datatype ,....., constraint pkey primary key(attribute1)) Create tablename (attribute1 datatype, attribute2 datatype ,....., constraint constraint_name primarykey(attribute1,attribute2))



Self-Activity
Steps to Use DDL statements

1. Login to linux server
2. Type the connection string to connect to database
psql -h IP address of server -d database-name
3. Type in the DDL statement at the sql> prompt

1. Type \h and go through the commands listed.
2. Type \h command-name & read through the help data given for each command.

Type the following Create table Statements to create the tables . If the table creation is successful you get 'create table' as output.

Then Type \d <table name> and write the output

3. Create table emp (eno integer primary key, ename varchar[50] , salary float);
4. Create table books(id integer UNIQUE, title text NOT NULL, author_id integer,sub_id integer,CONSTRAINT books_id_pkey PRIMARY KEY(id));
5. Create table sales_order(order_no char[10] PRIMARY KEY, order_date date, salesman_no integer);
6. Create table client_master(client_no integer CONSTRAINT p_client PRIMARY KEY, name varchar[50], addr text, bal_due integer);
7. Create table inventory(inv_no integer PRIMARY KEY,in_stock Boolean);

8. create table sales_order1(order_no char[10], product_no char[10], ,
 qty_ordered integer,product_rate numeric(8,2),PRIMARY
 KEY(order_no,product_no));

Signature of the instructor

Date / /



Set A

1.

Create a table with details as given below

Table Name	PLAYER		
Columns	Column Name	Column Data Type	Constraints
1	player_id	Integer	Primary key
2	Name	varchar (50)	
3	Birth_date	date	
4	Birth_place	varchar(100)	
Table level constraint			

2.

Create a table with details as given below

Table Name	Student		
Columns	Column Name	Column Data Type	Constraints
1	Roll_no	integer	
2	Class	varchar (20)	
3	Weight	numeric (6,2)	
4	Height	numeric (6,2)	
Table level constraint		Roll_no and class as primary key	

3.

Create a table with details as given below

Table Name	Project		
Columns	Column Name	Column Data Type	Constraints
1	project_id	integer	Primary key
2	Project_name	varchar (20)	
3	Project_description	text	
4	Status	boolean	
Table level constraint			

4.

Create a table with details as given below

Table Name	Donor		
Columns	Column Name	Column Data Type	Constraints
1	Donor_no	integer	Primary key
2	Donor_name	varchar (50)	
3	Blood_group	Char(6)	
4	Last_date	date	
Table level constraint			

Set B

Create table for the information given below by choosing appropriate data types and also specifying proper primary key constraint on fields which are underlined

1. Property (property_id, property_desc , area, rate, agri_status)
2. Actor (actor_id, Actor_name, birth_date)

3. Movie(movie-no, name, release-year)

4. Hospital(hno,hname,hcity)

Signature of the instructor

Date

Set C

Create table for the information given below by choosing appropriate data types and also specifying proper primary key constraint on fields which are underlined

1. Table _____ (_____, _____, _____, _____,
Primary key : _____

Instructor should fill in the blanks with appropriate values.

Signature of the instructor

Date

Assignment Evaluation

Signature

0: Not done

2: Late Complete

4: Complete

1: Incomplete

3: Needs improvement

5: Well Done

Exercise 2

Start Date / /



To create more than one table, with referential integrity constraint, PK constraint.



You should read following topics before starting this exercise

1. Referential Integrity constraints, relationship types (1-1,1-m,m-1,m-m)
2. Handling relationships while converting relations into tables in RDB, so that there are no redundancies.



The integrity constraints help us to enforce business rules on data in the database. Once an integrity constraint is specified for a table or a set of tables, all data in the table always conforms to the rule specified by the integrity constraint.

Referential integrity constraints designates a column or grouping of columns in a table called child table as a foreign key and establishes a relationship between that foreign key and specified primary key of another table called parent table.

The following is the list of constraints that can be defined for enforcing the referential integrity constraint.

Constraint	Use	Syntax and Example
Primary key	Designates a column or combination of columns as primary key	PRIMARY KEY (columnname[,columnname])
Foreign key	designates a column or grouping of columns as a foreign key with a table constraint	FOREIGN KEY (columnname[,columnname])
References	Identifies the primary key that is referenced by a foreign key. If only parent table name is specified it automatically references primary key of parent table	columnname datatype REFERENCES tablename[(columnname[,columnname])]
On delete Cascade	The referential integrity is maintained by automatically removing dependent foreign key values when primary key is deleted	columnname datatype REFERENCES tablename[(columnname)][ON DELETE CASCADE]
On update set null	If set, makes the foreign key column NULL, whenever there is a change in the primary key value of the referred table	Constraint name foreign key column- name references referred-table(column- name) on update set null

Rules to Handle relationships , attributes , enhanced E-R concepts during table creation :

Name	Description	Handling	Example	Create statement
One-to-one	A member from	The key	Room & guest.	Create table

	an entity set is connected to atmost one member from the other entity set & vice-versa	attribute from anyone entity set goes to the other entity set (may be the entity set that has full participation in relation) , as a foreign key.	Room no is foreign key in guest relation.guest has full participation in relation.	room(rno int primary key, desc char(30)); Create table guest(gno int, name varchar(20), rno int references room unique);
One-to-many, many-to-one	A member from the entity set on the one side is connected to one or more members from the other entity set, but a member from the entity set on the many side , is connected to atmost one member of the entity set on one side.	The key attribute of the entity set on one side is put as foreign key in the entity set on the many side.	Department & employee. Here department is on the one side & employee is on the many side.	Create table dept(dno int primary key, dname char(20)); Create table emp(eno int primary key, name char(30), dno int references dept);
May-to-many	A member from one entity set connected to one or more members of the other entity set & vice-versa	A new relation is created that will contain the key attributes of both the participating entity sets.	Student & subject . a student can opt for many subjects & a subject has many students opting for it.	Create table student(sno int primary key, name varchar(20)); Create table subject(sbno int primary key, name varchar(20)); Create table st-sub(sno int references student, sbno int references subject, constraint pkey primary key(sno,sbno));
A multivalued attribute	an attribute having multiple values for each member of the entity set	A new relation is created , which will contain a place holder for the multivalued attribute and the key attributes of the entity set that contains the multivalued attribute	An employee having multiple contact numbers, like home phone, mobile number, office number etc. hence the phone-no attribute in employee relation becomes a multivalued attribute.	Create table emp(eno int primary key, name char(30)); Create table emp-ph(eno int references emp, phno int , constraint pkey primary key(eno,phno));

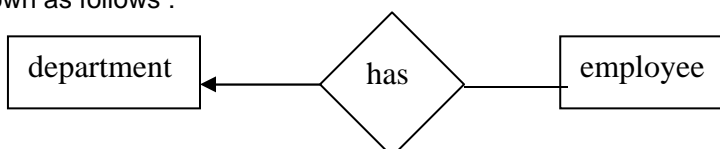
A multivalued, composite attribute	A composite attribute having multiple values , for each member of the entity set	A new relation is created, which will contain a place holder for each part of the composite attribute and the key attributes of the entity set that contains the composite multivalued attribute	An employee having multiple addresses , where each address is made up of a block no, street no, city, state. Hence the address attribute becomes a composite multivalued attribute.	Create table emp(eno int primary key, name char(30)); Create table emp-add(eno int references emp, addno int, hno int, street char(20), city char(20), constraint pkey primary key(eno,addno));
Generalization / specialization	The members of an entity set can be grouped into several subgroups, based on an attribute/s value/s. Each subgroup becomes an entity set. Depicts a parent-child type of relationship.	New relations for each subgroup , if the subgroups have its own attributes, other than the parent attributes. The parent entity set's key is added to each subgroup. If no specific attributes for each subgroup, then only the parent relation is created.	A person (parent entity set) can be an employee, a student, a retired person. Here employee has its own set of attributes like company, salary etc. a student has its own set of attributes like college/ school, course etc. a retired person has its own set of attributes like hobby, pension etc. so we create a person relation , a student relation, an employee, a retire person. The student , employee, retired person entity sets will have the key of the person entity set added to it.	Create table person(ssno int primary key, name char(30)); Create table emp(ssno int references person, eno int, cname char(20), sal float, primary key(ssno)); Create table student(ssno int references person, class char(10), school varchar(50), primary key(ssno));



Self-Activity

You can type the following Create table Statements to create the tables satisfying referential integrity constraints. On table creation type \d <table name> and write the output.

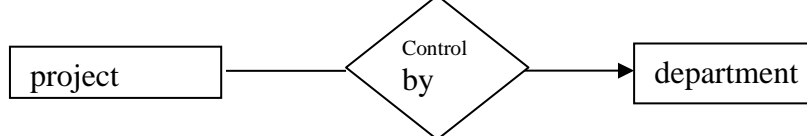
1. Consider two tables department & employee. One department can have one or more employees, but an employee belongs to exactly one department (1-m relation). It's pictorially shown as follows :



To handle the above relation, while creating the tables, 'deptno' is a foreign key in the employee table. The statement for creating the tables are as follows :

Create table department (deptno integer primary key, deptname text);
 Create table employee(empno integer primary key, ename varchar(50), salary float, dno integer references department(deptno) on delete cascade on update set null);

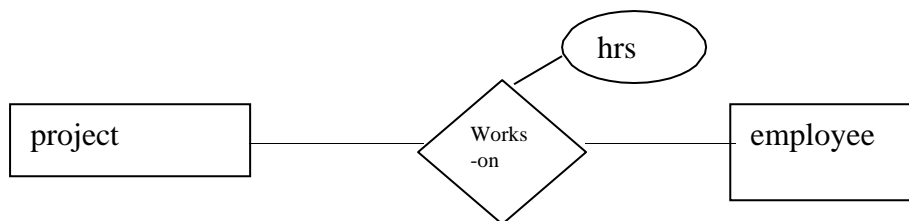
2. Consider the department table created above & another table called project. A project is controlled by exactly one department , but a department can control one or more projects(a m-1 relation). It's pictorially shown as follows :



To handle the above relationship, control-dno is a foreign key in project. The statement for creating the project table is as follows :

Create table project(pno int primary key, pname char(10), control-dno int, foreign key(control-dno) references department(dno))

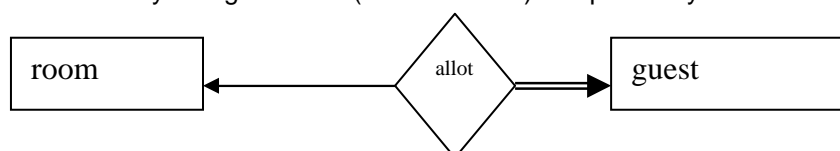
3. Consider the project & employee relations created above. An employee can work in one or more projects, and a project can have one or more employees working in it .(a m-m relation). It's shown pictorially as follows :



To handle the above relationship, we create a new table , works-on , as given below :

create table works(eno int references employee, pno int references project, hrs int, constraint pkey primary key(eno,pno))

4. Consider the relations guest and room. A guest is allocated exactly one room, and a room can contain exactly one guest in it. (a 1-1 relation). It's pictorially shown as follows :



To handle the above relation, we add room-no as foreign key to guest, since a guest cannot be without being allocated to a room (guest has full participation in relation). The statements for creating these relations are as follows

Create table room(room-no integer primary key , description char(20), charge integer);
 Create table guest(guest-no integer primary key, name varchar(30), room-no references room unique);

Signature of the instructor

Date / /



Set A

Create tables for the information given below by giving appropriate integrity constraints as specified.

1. Create the following tables :

Table Name		Property	
Columns	Column Name	Column Data Type	Constraints
1	Pnumber	Integer	Primary key
2	description	varchar (50)	Not null
3	Area	Char(10)	

Table Name		Owner	
Columns	Column Name	Column Data Type	Constraints
1	Owner-name	Varchar(50)	Primary key
2	Address	varchar (50)	
3	Phoneno	Integer	

Relationship → A one-many relationship between owner & property. Define reference keys accordingly .

2. Create the following tables :

Table Name		Hospital	
Columns	Column Name	Column Data Type	Constraints
1	Hno	Integer	Primary key
2	Name	varchar (50)	Not null
3	City	Char(10)	

Table Name		Doctor	
Columns	Column Name	Column Data Type	Constraints
1	Dno	Integer	Primary key
2	Dname	varchar (50)	
3	City	Char(10)	

Relationship → A many-many relationship between hospital & doctor.

3. Create the following tables :

Table Name		Patient	
Columns	Column Name	Column Data Type	Constraints
1	pno	Integer	Primary key
2	Name	varchar (50)	Not null
3	Address	Varchar(50)	

Table Name		Bed	
Columns	Column Name	Column Data Type	Constraints
1	Bedno	integer	Primary key
2	Roomno	integer	Primary key
3	description	Varchar(50)	

Relationship → a one-to-one relationship between patient & bed.

Signature of the instructor

Date / /

Set B

Create table for the information given below by choosing appropriate data types and integrity constraints as specified.

1. Table _____ (_____, _____, _____, _____
_____ (_____, _____, _____, _____
Constraints: _____, _____

2. Table _____ (_____, _____, _____, _____
_____ (_____, _____, _____, _____
Constraints: _____, _____

Relationship → _____

Instructor should fill in the blanks with appropriate values.

Signature of the instructor

Date

Assignment Evaluation

Signature

0: Not done

2: Late Complete

4: Complete

1: Incomplete

3: Needs improvement

5: Well Done

Exercise 3

Start Date



To create one or more tables with Check constraint , Unique constraint, Not null constraint , in addition to the first two constraints (PK & FK)



You should read following topics before starting this exercise

1. Different integrity constraints
2. Specification of different integrity constraints , in create table statement .



The following is the list of additional integrity constraints.

Constraint	Use	Syntax and Example
NULL	Specifies that the column can contain null values	CREATE TABLE client_master (client_no text NOT NULL , name text NOT NULL, addr text NULL , bal_due integer);
NOT NULL	Specifies that the column can not contain null values	CREATE TABLE client_master (client_no text NOT NULL , name text NOT NULL, addr text NOT NULL , bal_due integer);
UNIQUE	Forces the column to have unique values	CREATE TABLE client_master (client_no text UNIQUE, name text ,addr text, bal_due integer);
CHECK	Specifies a condition that each row in the table must satisfy. Condition is specified as a logical expression that evaluates either TRUE or FALSE.	CREATE TABLE client_master (client_no text CHECK(client_no like 'C%'), name text CHECK(name=upper(name)), addr text);



1. create client master by using any one DDL statement given above. On table creation type \d <table name> and write the output

Signature of the instructor

Date



Set A

1.

Create a table with details as given below

Table Name		Machine	
Columns	Column Name	Column Data Type	Constraints
1	Machine_id	integer	Primary key
2	Machine_name	varchar (50)	NOT NULL, uppercase
3	Machine_type	varchar(10)	Type in ('drilling', 'milling', 'lathe', 'turning', 'grinding')
4	Machine_price	float	Greater than zero
5	Machine_cost	float	
Table level constraint		Machine_cost less than Machine_price	

2.

Create a table with details as given below

Table Name		Employee	
Columns	Column Name	Column Data Type	Constraints
1	Employee_id	integer	Primary key
2	Employee_name	varchar (50)	NOT NULL, uppercase
3	Employee_desg	varchar(10)	Designation in ('Manager', 'staff', 'worker')
4	Employee_sal	float	Greater than zero
5	Employee_uid	text	Unique
Table level constraint		Employee_uid not equal to Employee_id	

Signature of the instructor

Date / /

Set B

1.

Create a table with details as given below

Table Name			
Columns	Column Name	Column Data Type	Constraints
1			
2			
3			
4			
5			
Table level constraint			

Instructor should fill in the blanks with appropriate values.

Signature of the instructor

Date / /

Assignment Evaluation

Signature

0: Not done 2: Late Complete 4: Complete
 1: Incomplete 3: Needs improvement 5: Well Done

Exercise 4-a

Start Date / /



To drop a table from the database, to alter the schema of a table in the Database.



You should read following topics before starting this exercise
1 Read through the drop, Alter DDL statement



The Drop & Alter DDL statements :

Name	Description	Syntax	Example
Drop	Deletes an object (table/view/constraint) schema from the database	Drop object-type object-name;	Drop table employee; Drop table sales; Drop constraint pkey;
Alter	<p>ALTER TABLE command of SQL is used to modify the structure of the table It can be used for following purposes</p> <p>a) adding new column b) modifying existing columns c) add an integrity constraint d) To redefine a column</p> <p>Restrictions on the alter table are that, the following tasks cannot be performed with this clause</p> <p>a) Change the name of the column b) Drop a column c) Decrease the size of a column if table data exists</p>	<p>Alter table tablename Add (new columnname datatype(size), new columnname datatype(size)...);</p> <p>Alter table tablename modify (columnname new datatype(new size),...);</p>	<p>Alter table emp(add primary key (eno)); alter table student(add constraint city- chk check city in (‘pune’, ‘mumbai’));</p> <p>alter table student modify (city varchar(100));</p>



Create the table given below . Assume appropriate data types for attributes.
Modify the table, as per the alter statements given below, type \d <table name>
and write the output.

Supplier_master(supplier_no, supplier_name,city,phone-no,amount)

1. Alter table supplier_master
add primary key (supplier_no);
2. Alter table supplier_master add constraint city-check check city in(‘pune’, ‘mumbai’, ‘calcutta’);
- 3.alter table supplier_master drop phone-no;

4. alter table supplier_master modify (supplier_name varchar(50));
5. alter table supplier_master drop constraint city-check;
6. drop table supplier_master;

Signature of the instructor

Date



Set A

1. Remove employee table from your database. Create table employee(eno, ename, sal). Add designation column in the employee table with values restricted to a set of values.
2. Remove student table from your database. Create table student(student_no, sname, date_of_birth). Add new column into student relation named address as a text data type with NOT NULL integrity constraint and a column phone of data type integer.
3. Consider the project relation created in the assignment 12. Add a constraint that the project name should always start with the letter 'R'
4. Consider the relation hospital created in assignment 12. Add a column hbudget of type int , with the constraint that budget of any hospital should always > 50000.

Signature of the instructor

Date

Set B

1. Remove _____ table from your database. Create table _____(_____ , _____, _____). Add _____

Instructor should fill in the blanks with appropriate values.

Signature of the instructor

Date

Assignment Evaluation

Signature

0: Not done	<input type="text"/>	2: Late Complete	<input type="text"/>	4: Complete	<input type="text"/>
1: Incomplete	<input type="text"/>	3: Needs improvement	<input type="text"/>	5: Well Done	<input type="text"/>

Exercise 4-b

Start Date / /



To insert / update / delete records using tables created in previous Assignments. (use simple forms of insert / update / delete statements)



You should read following topics before starting this exercise

1. Read through the insert , update, delete statement.
2. Go through the variations in syntaxes of the above statements.
3. Go through some examples of these statements
4. Go through the relationship types & conversion of relations to tables in RDB.
5. Normal forms → 1NF, 2NF, 3NF



The insert / update / delete statements

Name	Description	Syntax	Example
Insert	<p>The insert statement is used to insert tuples or records into a created table or a relation.</p> <p>We specify a list of comma-separated column values, which must be in the same order as the columns in the table.</p> <p>To insert character data we must enclose it in single quotes('). If a single quote is part of the string value to be inserted , then precede it with a backslash(\).</p> <p>When we don't have values for every column in the table, or the data given in insert is not in the same column order as in the table, then we specify the column names also alongwith the values (2nd syntax)</p>	<p>INSERT INTO tablename VALUES (list of column values);</p> <p>INSERT INTO tablename(list of column names) VALUES (list of column values corresponding to the column names);</p>	<p>Insert into emp values(1,'joshi',4000,'pune);</p> <p>Insert into emp(eno,city) values(2,'mumbai');</p>

Update	The UPDATE command is used to change or modify data values in a table. To specify update of several columns at the same time, we simply specify them as a comma-separated list	UPDATE tablename SET columnname = value where condition;	Update emp set sal = sal +0.5*sal; Update emp set sal = sal+1000 where eno =1;
Delete	The DELETE statement is used to remove data from tables.	DELETE FROM table name where condition;	Delete from emp ; Delete from emp where eno = 1;

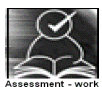


Consider the tables created in assignments 11,12,13,14. type and execute the below statements for these tables. Write the output of each statement & justify your answer

1. INSERT INTO sales_order(s_order_no,s_order_date,client_no)
VALUES ('A2100', now() , 'C40014');
2. INSERT INTO client_master values('A2100','NAVEEN','Natraj apt','pune_nagar road','pune',40014);
3. insert into client_master values ('A2100','NAVEEN',NULL,'pune_nagar road','pune',40014);
4. UPDATE emp SET netsal= net_sal_basic_sal*0.15;
5. UPDATE student
SET name='SONALI',address='Jayraj apartment'
WHERE stud_id=104 ;
6. DELETE from emp;
- 7.DELETE from emp
WHERE net_sal <1000;

Signature of the instructor

Date

 / /


Set A

1. Create the following tables (primary keys are underlined.).
Property(pno,description, area)
Owner(oname,address,phone)

An owner can have one or more properties, but a property belongs to exactly one owner . Create the relations accordingly ,so that the relationship is handled properly and the relations are in normalized form (3NF).

- a) Insert two records into owner table.
- b) insert 2 property records for each owner .
- c) Update phone no of "Mr. Nene" to 9890278008
- d) Delete all properties from "pune" owned by " Mr. Joshi"

- 2 . Create the following tables (primary keys are underlined.).
Emp(eno,ename,designation,sal)
Dept(dno,dname,dloc)

There exists a one-to-many relationship between emp & dept.

Create the Relations accordingly, so that the relationship is handled properly and the relations are in normalized form (3NF).

- a) Insert 5 records into department table
- b) Insert 2 employee records for each department.
- c) increase salary of "managers" by 15%;
- d) delete all employees of department 30;
- e) delete all employees who are working as a "clerk"
- f) change location of department 20 to 'KOLKATA'

3 . Create the following tables (primary keys are underlined.).

Sales_order(s_order_no,s_order_date)

Client(client_no, name, address)

A client can give one or more sales_orders , but a sales_order belongs to exactly one client. Create the Relations accordingly, so that the relationship is handled properly and the relations are in normalized form (3NF).

- a) insert 2 client records into client table
- b) insert 3 sales records for each client
- c) change order date of client_no 'C004' to 12/4/08
- d) delete all sale records having order date before 10th feb. 08
- e) delete the record of client "joshi"

Signature of the instructor

Date

Set B

1. Design a set of tables with the following constraints. Add any new attributes , as required by the queries.

Table name :

Field name	Data Type	Constraints

Table name :

Field name	Data Type	Constraints

Relationship → _____

Write & execute insert/ update / delete statements for following business tasks

- a)
- b)
- c)
- d)
- e)
- f)

Instructor should fill in the blanks with appropriate values.

Signature of the instructor

Date

Set C

Create an appropriate set of tables in normalized form to keep some business information. Populate the tables with information for the business process. State the updations that can be done to the data in the table .Write and execute update / delete statements for the same. The names of tables & fields should be self-explanatory (i.e . their names should depict the kind of data being stored.)

Signature of the instructor

Date

Assignment Evaluation

Signature

0: Not done

2: Late Complete

4: Complete

1: Incomplete

3: Needs improvement

5: Well Done

Exercise 5

Start Date / /



To understand & get a Hands-on on Select statement



- You should read following topics before starting this exercise
1. Creating relations as tables and inserting tuples as records into the table.
 2. The use of select statement in extracting information from the relation.
 3. Insert/update/delete with subquery.
 4. Relationship types & their conversion to RDB.
 5. Normal forms → 1NF, 2NF, 3NF.



The select statement :

Name	Description	Syntax	Example
Select statement	<p>Used to read a tuple, tuples, parts of a tuple from a relation in the database. Tuple means a record in an RDB & a relation means a table.</p> <p>The basic structure of a Select statement consists of 3 clauses :</p> <p>The select clause → corresponds to the projection operation in relational algebra. It is used to list the attributes desired in the query.</p> <p>The from clause → corresponds to the Cartesian product operation of RA. It lists the relations to be scanned in the evaluation of the expression.</p> <p>The where clause → corresponds to the selection operation of RA. It consists of a predicate involving the attributes of the relations that appear in the select clause.</p> <p>The other clauses are :</p> <p>Order by clause → causes the result of the</p>	<pre>select <attribute-list> from <relation-list> [where <condition> [group by <attribute list> [having <condition>] order by <attribute list>]];</pre>	<pre>Select * from emp; Select eno,name from emp; Select eno name from emp where sal > 4000 order by eno; Select sum(sal) from emp group by dno having sum(sal)> 100000;</pre>

	<p>query to appear in a sorted order.</p> <p>Group by clause → used to form groups of tuples , of the result . It is used when using aggregate functions.</p> <p>Having clause → Used with group by clause, to force a condition on the groups formed after applying group by clause, & selects only those groups in the output that satisfy the condition.</p> <p>The order of execution of the clauses is the same as given in the syntax.</p>		
--	--	--	--

The aggregate functions supported by SQL are as follows:

Name	Description	Usage	Example
Sum()	Gets the sum or total of the values of the specified attribute.	Sum(attribute-name)	Select sum(sal) from emp; Select dno, sum(sal) from emp group by dno;
Count()	Gives the count of members in the group	Count(attribute); Count(*)	Select count(*) from emp; Select count(*) from emp where sal > 5000;
Max()	Gives the maximum value for an attribute, from a group of members	Max(attribute)	Select max(sal) from emp; Select dno, max(sal) from emp group by dno having count(*) >10;
Min()	Gives the minimum value for an attribute, from a group of members	Min(attribute)	Select min(sal) from emp; Select dno, min(sal) from emp group by dno having min(sal) >100;
Avg()	Gives the average value for an attribute, from a group of members	Avg(attribute)	Select avg(sal) from emp; Select dno, avg(sal) from emp group by dno having count(*) >10;



As part of the self activity in exercise you have created a table employee with attributes empno, name, address, salary and deptno. You have also inserted atleast 10 records into the same.

To execute each query

type each query into the database prompt or

type queries in a file and cut and copy each query at the database prompt or

type queries in a file and type \i filename at SQL prompt. (all queries in the file will get executed one by one).

Execute following select queries & write the business task performed by each query.

1. Select * from emp;
2. Select empno, name from emp;
3. Select distinct deptno from emp;
4. Select * from emp where deptno = ___;
5. Select * from emp where address = 'pune' and sal > _____;
6. Select * from emp where address = 'pune and salary between _____ and _____;
7. Select * from emp where name like '---%'
8. Select * from emp where name like '%and%'
9. Select * from emp where salary is null;
10. Select * from emp order by eno;
11. Select * from emp order by deptno, eno desc;
12. Select deptno as department, sum(salary) as total from emp group by deptno order by deptno;
13. Select deptno as department , count(eno) as total_emp from emp group by deptno having count(eno) > _____ order by deptno;
14. select avg(salary) from emp;
15. select max(salary),deptno from emp group by deptno having max(sal) > _____;
16. select deptno, min(salary) from emp order by deptno;
17. update emp set salary = salary + 0.5*salary where deptno = (select deptno from department where dname = 'finance');
18. update emp set deptno = (select deptno from department where dname = 'finance')
Where deptno = (select deptno from department where dname = 'inventory');
19. insert into emp_backup(eno,ename) values(select eno,ename from emp);
20. delete from emp where deptno = (select deptno from department where dname='production');

Signature of the instructor

Date



Set A

Consider the relations Person (pnumber, pname, birthdate, income), Area(aname,area_type). An area can have one or more person living in it , but a person belongs to exactly one area. The attribute 'area_type' can have values as either urban or rural.

Create the Relations accordingly, so that the relationship is handled properly and the relations are in normalized form (3NF).

Assume appropriate data types for all the attributes. Add any new attributes as required, depending on the queries. Insert sufficient number of records in the relations / tables with appropriate values as suggested by some of the queries.

Write select queries for following business tasks and execute them.

1. List the names of all people living in '_____' area.
2. List details of all people whose names start with the alphabet '_' & contains maximum _ alphabets in it.
3. List the names of all people whose birthday falls in the month of _____.
4. Give the count of people who are born on '_____'
5. Give the count of people whose income is below _____.
6. List names of all people whose income is between _____ and _____;

7. List the names of people with average income
8. List the sum of incomes of people living in '_____'
9. List the names of the areas having people with maximum income (duplicate areas must be omitted in the result)
10. Give the count of people in each area
11. List the details of people living in '_____ ' and having income greater than _____;
12. List the details pf people, sorted by person number
13. List the details of people, sorted by area, person name
14. List the minimum income of people.
15. Transfer all people living in 'pune' to 'mumbai'.
16. delete information of all people staying in 'urban' area

Instructor should fill in the blanks with appropriate values.

Signature of the instructor

Date

Set B

1. Design a table with the following constraints. Add any new attributes , as required by the queries.

Table name :

Field name	Data Type	Constraints

Insert sufficient number of records in the relations / tables with appropriate values as suggested by some of the queries.

Write & execute select queries for following business tasks

- a)
- b)
- c)
- d)
- e)

Instructor should fill in the blanks with appropriate values.

Signature of the instructor

Date

Set C

Create an appropriate table to keep some business information. Populate the table with information for the business process. State the business tasks that you need to perform to extract information. Write and execute queries for the same. The names of tables & fields should be self-explanatory (i.e. their names should depict the kind of data being stored.)

Signature of the instructor

Date

Assignment Evaluation

Signature

- | | | |
|------------------------------------|---|-----------------------------------|
| 0: Not done <input type="text"/> | 2: Late Complete <input type="text"/> | 4: Complete <input type="text"/> |
| 1: Incomplete <input type="text"/> | 3: Needs improvement <input type="text"/> | 5: Well Done <input type="text"/> |

Exercise 6

Start Date / /



To understand & get a Hands-on on using set operations (union ,intersect and except) with select statement.



You should read following topics before starting this exercise

1. Relation algebra operation \cap , \cup and $-$.
2. SQL operations union, intersect & except



SQL Set operations :

Name	description	Syntax	Example
Union	Returns the union of two sets of values, eliminating duplicates.	<select query> Union <select query>	Select cname from depositor Union Select cname from borrower;
Union all	Returns the union of two sets of values, retaining all duplicates.	<select query> Union all <select query>	Select cname from depositor Union all Select cname from borrower;
Intersect	Returns the intersection of two sets of values, eliminating duplicates	<select query> intersect <select query>	Select cname from depositor intersect Select cname from borrower;
Intersect all	Returns the intersection of two sets of values, retaining duplicates	<select query> Intersect all <select query>	Select cname from depositor Intersect all Select cname from borrower;
except	Returns the difference between two set of values, i.e returns all values from set1 , not contained in set2 .eliminates duplicates	<select query> except <select query>	Select cname from depositor except Select cname from borrower;
Except all	Returns the difference between two set of values, i.e. returns all values from set1 , not contained in set2 .Retains all duplicates	<select query> Except all <select query>	Select cname from depositor Except all Select cname from borrower;

The relations participating in the SQL operations union, intersect & except must be compatible i.e. the following two conditions must hold :

- a) The relation r and s must be of the same arity. That is , they must have the same number of attributes.

b) The domains of the i^{th} attribute of r and the i^{th} attribute of s must be the same , for all i .

Consider the following relations, non-teaching, teaching, department.
 One department can have one or more teaching & non-teaching staff, but a teaching or non-teaching staff belongs to exactly one department. Hence dno is a foreign key in the both the relations. Create these relations in your database .

Non-teaching (empno int primary key, name varchar(20), address varchar(20), salary int,dno references department)

Teaching(empno int primary key, name varchar(20), address varchar(20), salary int,dno references department)

Department(dno int primary key,dname)

- insert at least 10 records into both the relations.
- type the following select queries & write the output and the business task performed by each query

1. Select empno from non-teaching union select empno from teaching;
2. Select empno from non-teaching union all select empno from teaching;
3. Select name from non-teaching intersect select name from teaching;
4. Select name from non-teaching intersect all select name from teaching;
5. Select name from non-teaching except select name from teaching;
6. Select name from non-teaching except all select name from teaching;

Signature of the instructor

Date



Set A

Create the following relations, for an investment firm
 emp(emp-id ,emp-name, address, bdate)
 Investor(inv-name , inv-no, inv-date, inv-amt)

An employee may invest in one or more investments, hence he can be an investor.
 But an investor need not be an employee of the firm.

Create the Relations accordingly, so that the relationship is handled properly and the relations are in normalized form (3NF).

Assume appropriate data types for the attributes. Add any new attributes , as required by the queries. Insert sufficient number of records in the relations / tables with appropriate values as suggested by some of the queries.

Write the following queries & execute them.

1. List the distinct names of customers who are either employees, or investors or both.
2. List the names of customers who are either employees , or investors or both.
3. List the names of employees who are also investors.
4. List the names of employees who are not investors.

Signature of the instructor

Date

Set B

1. Design following two tables with the following constraints . Add any new attributes, as required by the queries.

Table name 1:

Field name	Data Type	Constraints

Table name 2:

Field name	Data Type	Constraints

Relationship → _____

Insert sufficient number of records in the relations / tables with appropriate values as suggested by some of the queries.

Write & execute queries for following business tasks

- a)
- b)
- c)
- d)
- e)
- f)

Instructor should fill in the blanks with appropriate values.

Signature of the instructor

Date

Set C

Create two compatible tables having similar set of attributes, to keep some business information. Populate the tables with information for the business process. State the business tasks that you need to perform on these tables involving information from both the tables. Write and execute queries for the same. The names of tables & fields should be self-explanatory (i.e. their names should depict the kind of data being stored.)

Signature of the instructor

Date

Assignment Evaluation

Signature

0: Not done	<input type="text"/>	2: Late Complete	<input type="text"/>	4: Complete	<input type="text"/>
1: Incomplete	<input type="text"/>	3: Needs improvement	<input type="text"/>	5: Well Done	<input type="text"/>

Exercise7-a

Start Date / /



To understand & get a Hands-on on nested queries & subqueries, that involves joining of tables.



You should read following topics before starting this exercise

1. Nesting of SQL queries and subqueries
2. SQL statements involving set membership, set comparisons and set cardinality operations.
3. Descriptive attributes & how they are handled while creating RDB.



A subquery is a select-from-where expression that is nested within another query.

Set membership	the 'in' & 'not in' connectivity tests for set membership & absence of set membership respectively.
Set comparison	the < some, > some, <= some, >= some, = some, <> some are the constructs allowed for comparison. = some is same as the 'in' connectivity. <> some is not the same as the 'not in' connectivity. Similarly sql also provides < all, >all, <=all, >= all, <> all comparisons. <>all is same as the 'not in' construct.
Set cardinality	The 'exists' construct returns the value true if the argument subquery is nonempty. We can test for the non-existence of tuples in a subquery by using the 'not exists' construct. The 'not exists' construct can also be used to simulate the set containment operation (the super set). We can write "relation A contains relation B" as "not exists (B except A)".

The complete Syntax of select statement containing connectivity or Comparison operators is as follows

```
select <attribute-list> from <relation-list>
      where <connectivity / comparison > { sub-query };
```



Create the following relation in your database(primary keys underlined)

- Employee(ename, street, city)
- Works(ename, company-name, salary)
- Company(company-name, city)
- Manages(ename, manager-name)

An employee can work in one or more companies, a company can have one or more employees working in it. Hence the relation 'works' with key attributes as ename, company-name.

An employee manages one or more employees, but an employee is managed by exactly one employee (a recursive relationship), hence the relation 'manages' with key ename.

Insert sufficient number of records in the relations / tables with appropriate values as suggested by some of the queries.

Type the following queries , execute them and give the business task performed by each query

1. select ename from works w where salary >= all (select max(salary) from works);
2. select ename form works w where salary = (select max(salary) from works w1 where w1.company-name = w.company-name));
3. select manager-name from manages where manager-name in(select ename from works where company-name = "_____");
4. select manager-name from manages where manager-name not in(select ename from works where company-name = "_____");

5. select ename from works w where salary > some (select salary from works where company-name not in (select company-name from company where city = "_____"));
6. select ename from employee e where city = (select city from employee e1 , manages m where m.ename = e.ename and m.manager-name = e1.ename);
7. select * from employee where ename in (select manager-name from manages)
8. select city count(*) from employee group by city having count(*) >= all (select count(*) from employee group by city)
9. select ename from works w where salary <> all (select salary from works where ename <> w.ename);
10. select company-name, sum(salary) from works w group by company-name having sum(sal) >= all (select sum(sal) from works group by company-name)
11. select ename from employee e where city in('_____', '_____');
12. select ename from employee e where city = (select city from company c, works w where w.ename = e.name and c.company-name = w.company-name);

Instructor should fill in the blanks with appropriate values.

Signature of the instructor

Date



Set A

Create the following relations :

Emp(eno,name,dno,salary)

Project(pno,pname,control-dno,budget)

Each employee can work on one or more projects, and a project can have many employees working in it. The number of hours worked on each project , by an employee also needs to be stored.

Create the Relations accordingly, so that the relationship is handled properly and the relations are in normalized form (3NF).

Assume appropriate data types for the attributes. Add any new attributes , new relations as required by the queries.

Insert sufficient number of records in the relations / tables with appropriate values as suggested by some of the queries.

Write the queries for following business tasks & execute them.

1. list the names of departments that controls projects whose budget is greater than_____.
2. list the names of projects, controlled by department No __, whose budget is greater than atleast one project controlled by department No_____.
3. list the details of the projects with second maximum budget
4. list the details of the projects with third maximum budget.
5. list the names of employees, working on some projects that employee number_____is working.
6. list the names of employees who do not work on any project that employee number_____works on
7. list the names of employees who do not work on any project controlled by '_____' department
8. list the names of projects along with the controlling department name, for those projects which has atleast____employees working on it.
9. list the names of employees who is worked for more than 10 hrs on atleast one project controlled by '_____' dept.
10. list the names of employees , who are males , and earning the maximum salary in their department.
11. list the names of employees who work in the same department as '_____ '.
12. list the names of employees who do not live in_____or_____.

Instructor should fill in the blanks with appropriate values.

Signature of the instructor

Date

Set B

1. Design a set of tables with the following constraints. Add any new attributes , as required by the queries.

Table name :

Field name	Data Type	Constraints

Table name :

Field name	Data Type	Constraints

Table name :

Field name	Data Type	Constraints

Relationship → _____

Insert sufficient number of records in the relations / tables with appropriate values as suggested by some of the queries.

Write & execute queries for following business tasks. (business tasks should be using set operations & should be similar to the ones given in set A)

- a)
- b)
- c)
- d)

Instructor should fill in the blanks with appropriate values.

Signature of the instructor

Date

Set C

Create an appropriate set of tables to keep some business information. Populate the tables with information for the business process. State the business tasks that involve set of operations that you need to perform to extract information. Write and execute queries for the same. The names of tables & fields should be self-explanatory (i.e. their names should depict the kind of data being stored.)

Signature of the instructor

Date

Assignment Evaluation

Signature

- | | | |
|------------------------------------|---|-----------------------------------|
| 0: Not done <input type="text"/> | 2: Late Complete <input type="text"/> | 4: Complete <input type="text"/> |
| 1: Incomplete <input type="text"/> | 3: Needs improvement <input type="text"/> | 5: Well Done <input type="text"/> |

Exercise 7-b

Start Date / /



To understand & get a Hands-on on nested queries & subqueries, that involves joining of tables, to demonstrate set cardinality.



You should read following topics before starting this exercise

1. Nesting of SQL queries and subqueries
2. SQL statements involving set membership, set comparisons and set cardinality operations.



SQL includes a feature for testing whether a subquery has any tuples in its result, using the following clauses .:

Name	Description	Syntax	Example
Exists	The 'exists' construct returns the value true if the argument subquery is nonempty	select <attribute-list> from <relation-list> where <exists> { sub-query } ;	Select cname from borrower b where exists(select * from depositor where dname = b.cname);
Not exists	We can test for the non-existence of tuples in a subquery by using the 'not exists' construct. The 'not exists ' construct can also be used to simulate the set containment operation (the super set). We can write "relation A contains relation B" as "not exists (B except A)"	select <attribute-list> from <relation-list> where <not exists> { sub-query};	Select cname from borrower b where not exists(select * from depositor where dname = b.cname);



Consider the table you have prepared as part of self activity of exercise 18, Type the following queries , execute them and give the business task performed by each query

1. Select company-name from company c where not exists (select city from company where company-name = "_____" except (select city from company where company-name = c.company-name));
2. Select ename from employee e where exists (select manager-name from manages where manager-name = e.ename group by manager-name having count(*) >3);
3. Select company-name from company c where not exists (select city from company where company-name = c.company-name except (select city from company where company-name = "____"));
4. Select ename from employee e where exists (select city from employee where city = e.city and ename <> e.ename group by city having count(*) > 5)

5. Select company-name from company c where not exists (select company-name from company where city = c.city and company-name <> c.company-name)

Instructor should fill in the blanks with appropriate values.

Signature of the instructor

Date



Set A

Consider the table you have prepared as part of Assessment work set A of exercise 18, Type the following queries, execute them and give the business task performed by each query

1. List the names of employees who work in all the projects that “_____” works on.
2. List the names of employees who work on only some projects that “_____” works on
3. List the names of the departments that have atleast one project under them .(write using ‘exists ‘ clause)
4. List the names of employees who do not work on “sales” project (write using ‘not exists’) clause
5. List the names of employees who work only on those projects that are controlled by their department .
6. List the names of employees who do not work on any projects that are controlled by their department

Instructor should fill in the blanks with appropriate values.

Signature of the instructor

Date

Set B

1. Design tables with the following constraints. Add any new attributes , as required by the queries.

Table name :

Field name	Data Type	Constraints

Table name :

Field name	Data Type	Constraints

Relationship → _____

Insert sufficient number of records in the relations / tables with appropriate values as suggested by some of the queries.

Write & execute queries for following business tasks. (business tasks should be using set cardinality operations & should be similar to the ones given in set A)

a)

b)

- c)
- d)
- e)
- f)

Instructor should fill in the blanks with appropriate values.

Signature of the instructor

Date

Set C

Create an appropriate set of tables to keep some business information. Populate the tables with information for the business process. State the business tasks that involve set cardinality operations that you need to perform to extract information. Write and execute queries for the same. The names of tables & fields should be self-explanatory (i.e. their names should depict the kind of data being stored.)

Signature of the instructor

Date

Assignment Evaluation

Signature

0: Not done	<input type="text"/>	2: Late Complete	<input type="text"/>	4: Complete	<input type="text"/>
1: Incomplete	<input type="text"/>	3: Needs improvement	<input type="text"/>	5: Well Done	<input type="text"/>

Exercise8

Start Date



Assignment to create views

