



Savitribai Phule Pune University,Pune

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

CS113 : Practical course on Problem Solving using
Computer and 'C' programming (CS111) and Database
Management Systems(CS112)

WorkBook

(From Academic year 2019)

Name : _____

College Name : _____

Roll No. : _____ Division : _____

Academic Year : _____

Preface to the Second Edition

The new syllabus for F.Y.B.Sc. Computer Science is implemented from the academic year 2019. For the subject CS113: Practical course based on CS111 and CS112.

It is absolutely necessary and essential that all the Computer Science practicals be conducted on Open Source Operating System like Linux. All the practicals related to C needs to be conducted using GCC compiler.

The practical examination will be conducted by the respective colleges at the end of the academic year, 35 marks will be assigned to performance in practical examination and 15 marks for journals through continuous assessment.

It is mandatory to carry the completed and duly signed lab book for the practical examination.

Editors:

Prof. Ms. Poonam Ponde	Nowrosjee Wadia College
Prof. Ms. Ashwini Kamble	Nowrosjee Wadia College

Prof. Ms. Vandana Babrekar	Nowrosjee Wadia College
Prof. Ms. Rashmi Baghel	Nowrosjee Wadia College

Updated By

Ms.Sunita N.Deore	K.T.H.M.College, Nashik
Ms.Ratna S.Chaudhari	K.T.H.M.College, Nashik
Ms.Asmita L.Taskar	K.T.H.M.College, Nashik
Ms.Archana R.Mogal	K.T.H.M.College, Nashik

Introduction

1. About the work book

This workbook is intended to be used by F.Y.B.Sc. (Computer Science) students for the 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) Carry this book every time you come to the lab for computer science practicals
- 2) A notebook 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 beforehand for the Exercise by reading the material mentioned under.
- 4) If the self-activity exercise or assessment work contains any blanks such as _____, get them filled by your instructor.
- 5) Instructor will specify which problems you are to solve by ticking box
- 6) 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

2.2. 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) You should evaluate each assignment carried out by a student on a scale of 5 as specified above by ticking appropriate box.
- 7) The value should also be entered on assignment completion page of the respective Lab course

2.3. Instructions to the Lab administrator

You have to ensure appropriate hardware and software is made 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 *
 - b. Servers Side (software's to be installed)
In Linux – C, C++, awk, shell, perl, postgresql/MySQL
- 2) Client Side (Operating System)
 - * Red Hat Linux and Fedora Core *
 - Client Side (software's to be installed)
In Linux – C, C++, awk, shell, perl, postgresql/mysql

Information about installation and configuring of the server and client are provided in appendix A

First Edition Editors:

Dr. Mrs. S. C. Shirwaikar
Prof. Poonam Ponde
Prof. Reena Bharathi
Prof. Manisha Suryavanshi
Prof. Jeevan Limaye
Prof. Kalyani Ghanwat
Prof. Anagha Joshi
Prof. Vandana Babrekar
Prof. Manasi Keskar
Prof. Seema Purandare Prof.
Padmavathy M.
Prof. Anjali Sardesai
Prof. Parag Tamhankar
Prof. Rasika Rahalkar

Assignment Completion Sheet

A) Problem Solving and C programming		
Sr.No	Assignment Name	Marks
1	Problem Solving using Pseudo code and Flowchart, Simple programs, Understanding errors and error handling.	
2	Decision Making Control Structures.	
3	Loop Control Structures	
4	Functions (User Defined functions, Library functions and Recursion).	
5	Arrays (1-D and 2-D)	

B) Database Management Systems		
Sr.No	Assignment Name	Marks
1	To create simple tables with only the primary key constraint (as a table level constraint & as a field level constraint) (include all data types)	
2	To create more than one table, with referential integrity constraint, PK constraint	
3	To create one or more tables with following constraints, in addition to the first two constraints (PK & FK) a. Check constraint b. Unique constraint c. Not null constraint	
4	To drop a table, alter schema of a table, insert / update / delete records using tables created in previous Assignments. (use simple forms of insert / update / delete statements)	
5	To query the tables using simple form of select statement Select from table [where order by] Select from table [where group by <> having <> order by <>]	
6	To query table, using set operations (union, intersect)	
7	To query tables using nested queries (use of 'Except', exists, not exists, all clauses	
8	To create views.	

CERTIFICATE

This is to certify that, Mr./Miss. _____ Roll No. _____ of
 F.Y.B.Sc. (Computer Science) has successfully completed _____ out of _____ practicals satisfactorily
 during the academic year 20 - 20 .

Practical In-charge

Head,
 Dept.Of Computer Science

Internal Examiner

External Examiner



Savitribai Phule Pune University,Pune

CS113: Practical course on Problem Solving using
Computer and ‘C’ programming (CS111) and Database
Management Systems (CS112)

Section A

Problem Solving using Computer and ‘C’ programming

You should read following topics before starting this exercise

1. UNIX and LINUX operating system

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 needs 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

Assignment 1

Date:

Problem Solving using Pseudo code and Flowchart, Simple programs, understanding errors and error handling.

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

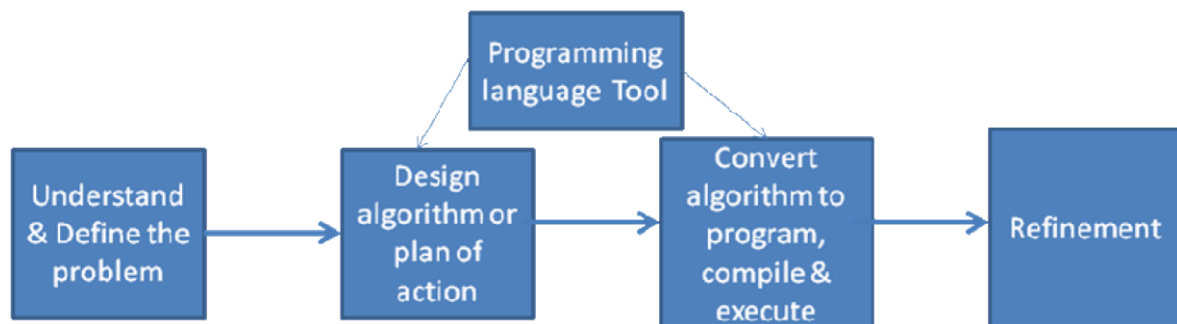
Reading-

You should read following topics before starting this exercise

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

Program solving using Computers

The steps in solving a problem using computers are shown below



Pseudo Language conventions

1. Appropriate names should be given to variables – the algorithm itself is given a name

Algorithm FindingMaximum

2. Organized sequence of data values is given a name and each data value is indicated by giving the index in brackets

Values[position]

3. Expressions containing Arithmetic operators +, -, *, / relational operators <, ≤, >, ≥, ≠, = and logical and, or, not can be used

Example: length * breadth, previous < next

4. Assignment statement variable = expression is used to assign data values to variables
5. Conditional constructs for making decisions

if condition then statements

Statements are to be carried out only if condition is true

if condition then statements1 else statements2

Statements1 are to be carried out only if condition is true and statements2 are to be carried out if condition is false

6. Loops

- i. while condition do statements

Statements are to be repeated while condition is true

- ii. for variable = value1 to value2 do statements

Statements are to be repeated moving from value1 to value2, with an increment one step.

- iii. for variable = value1 downto value2 do { statements }

Statements are to be repeated moving from value1 down to value2, with a decrement one step.

- iv. repeat { statements } until condition

Statements are to be repeated until condition is true

7. Input /output statements

- i. read value For taking value as input for the algorithm

- ii. write value For printing value as output of the algorithm

8. Algorithm name(value1 , valu2) – taking input for the algorithm

9. return value1 , value2 – for giving output of the algorithm

Examples:

Problem Statement: Accept radius and calculate area and circumference of a circle

Algorithm AreaCircumference

Begin

Input radius pi=3.142

area = piXradiusXradius

circum= 2XpiXradius Output

area

Output circum

End

Problem Statement: Check if a number is even

Algorithm Even

Begin Input m if m mod 2 = 0 then

output “m is even”

End

Problem Statement: Find maximum of two numbers

Algorithm Maximum

Begin

Input m Input n
if
m > n then output
m
else output n

End

Problem Statement: Give a discount of 15 % when purchase amount exceeds 5000, otherwise give a discount of 10%

Algorithm Discount Input

amount

if amount exceeds 5000 then compute discount as 25 times
amount divided by 100

else compute discount as 15 times amount divided by 100

Subtract discount from amount

Output amount

End

Problem Statement: Given a set of 100 values representing marks of students, count the total students that have passed. (A score of 40 is required for passing.)

Algorithm PassCount1

Begin

count = 0

n = 1

While n <= 100 do

Input marks

If marks >= 40 then

increment count by 1

n = n+1

Output count

End

The same can be done using another loop construct as shown below:

Algorithm PassCount1

Begin

count = 0

for n = 1 to 100 do

Input marks

If marks >= 40 then increment

count by 1 Output count

End

Problem Statement: Accept characters till a * is entered from the keyboard and count the number of characters entered.

Algorithm CharacterCount

```

Begin
    count = 0
    Input char
    while char ≠ * do
count = count + 1
        Input char
    Output count
End
  
```


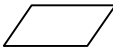

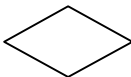

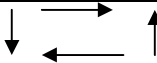
Problem Statement: Accept a number and calculate the sum of its digits.

Algorithm SumDigits

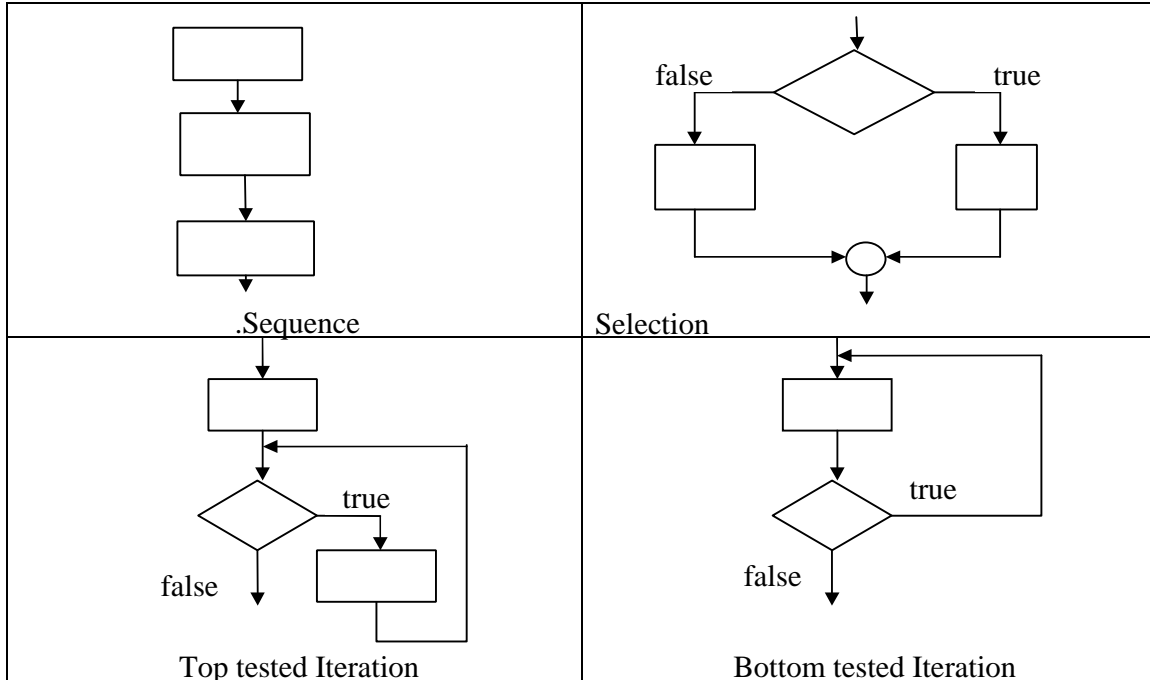
```

Begin
    Input num
    sum=0
    repeat
        digit = num mod 10
        sum=sum+digit
    num = num/10
    until num>0 Output
    sum
End
  
```

FLOWCHART SYMBOLS

Start Statement	
Input Statement	
Statement or procedure	
Decision, choice or Selection	
Connectors	
Control flow	

Basic control structures in an algorithm: Sequence, Selection and Iteration are shown below

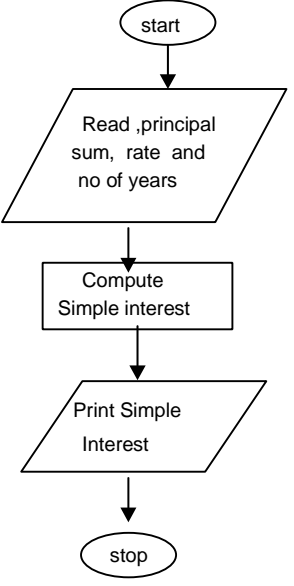


1. Data type Table

Data	Data Format	C Data Type	C Variable declaration	Input Statement	Output statement
quantity month creditcard 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("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'

Step 1: Writing the Pseudocode	Step 2: Draw the flowchart	Step 3: Write the Program
<p>Algorithm SimpleInterest</p> <p>Begin</p> <p>Input amount, rate, years</p> <p>si =amountXyearsXrate / 100</p> <p>Output si</p> <p>End</p>	 <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> #include <stdio.h> void 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>

Self Activity

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 gedit followed by filename. The filename should have .c as extension for example \$gedit pnr.c
 - b. Type the sample program 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. The program can be executed by typing name of the file as follow \$./a.out

Alternatively, the executable file can be given name by using -o option while compiling as follows \$cc

pnr.c -o pnrexec

\$/pnrexec

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

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 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.

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.

Assignment Evaluation

0: Not Done []

1: Incomplete []

2: Late Complete []

3: Needs Improvement []

4: Complete

5: Well Done []

Assignment 2

Date:

Decision Making Control Structures

Exercise 1:

Objective:

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

Reading:

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

Sr. No	Statement Syntax	Flowchart	Example
1.	if statement if (condition) { statement; }	<pre>graph TD; Start(()) --> Cond{If condition?}; Cond -- True --> S1[statement]; Cond -- False --> S2[New statement]; S1 --> S2; S2 --> End(())</pre>	<pre>if(n > 0) printf("Number is positive");</pre>
2.	if - else statement if (condition) { statement; } else { statement; }	<pre>graph TD; Start(()) --> Cond{If condition?}; Cond -- True --> S1[statement]; Cond -- False --> S2[statement]; S1 --> S3[New statement]; S2 --> S3; S3 --> End(())</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{a >= c} D3 -- True --> AMax D3 -- False --> CMax[c is max] 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
<p>Algorithm Range Begin Input llimit, ulimit Input num If $n \geq llimit$ and $n \leq ulimit$ then Output "in range" Else Output "not in range" End</p>	<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 of range/] W --> Stop((stop)) O --> Stop </pre>	<pre> #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>

Self Activity

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.

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

1. Write a program to accept an integer and check if it is even or odd.
2. 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
3. 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)
4. Write a program to accept a number and check if it is divisible by 5 and 7.
5. 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%

6. Accept a character from the user and check whether the character is a vowel or consonant. (Hint: a,e,i,o,u, A, E, I, O, U are vowels)
7. 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)

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 58 and Lowercase characters have ASCII values in the range of 97 to122, uppercase is between 65 and 90)
2. Accept the x and y coordinate of a point and find the quadrant in which the point lies.
3. Write a program to calculate the roots of a quadratic equation. Consider all possible cases. 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.
4. Write a program to accept marks for three subjects. Calculate the average and also display the class obtained. (Distinction – above _____ Class I – above __%, class II – ___% to __%, pass class – ___% to ___% and fail otherwise)

Assignment Evaluation

0: Not Done [] 1: Incomplete [] 2: Late Complete []
3: Needs Improvement [] 4: Complete 5: Well Done []

Date:

Exercise 2:

Objective:

To demonstrate decision making statements (switch case)

Reading:

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 switchcase 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 --> DefaultBlock[Default Block] Block1 --> DefaultBlock DefaultBlock --> Stop([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
<p>Algorithm Menu</p> <p>Begin</p> <p>Output menu</p> <p>Read choice</p> <p>Execute statements depending on choice</p> <p>End</p>	<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 -- False --> Default[Default statement] S1 --> Stop([stop]) S2 --> Stop S3 --> Stop Default --> Stop </pre>	<pre> #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>

Self Activity

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	
y	

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

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

Assignment Evaluation

0: Not Done []

1: Incomplete []

2: Late Complete []

3: Needs Improvement []

4: Complete

5: Well Done []

Assignment 3

Date:

Loop Control Structures

Exercise 1:

Objective:

To demonstrate use of simple loops.

Reading:

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

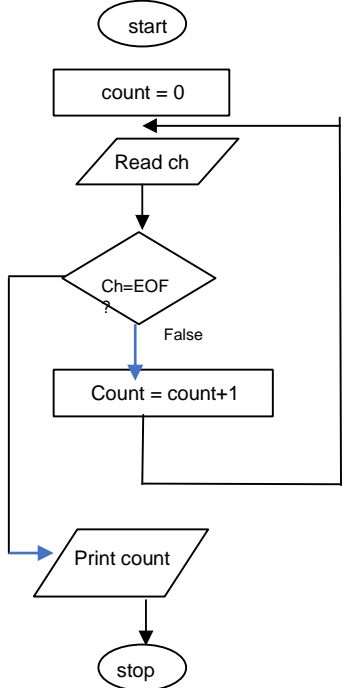
Sr. No	Statement Syntax	Flowchart	Example
1.	while statement <pre>while (condition) { statement 1; statement 2; . . }</pre>		<pre>/* accept a number*/ scanf("%d", &n); /* if not a single digit */ while (n > 9) { /* remove last digit n = n /10; }</pre>
2.	do-while statement <pre>do { statement 1; statement 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 <pre> for (expr1; expr2; expr3) { statement 1 : } </pre> <p> expr1 = initialization expression expr2 = loop condition expr3 = alteration expression which alters the loop variable </p>	<pre> graph TD Start(()) --> Expr1[expr1] Expr1 --> Test{Test expr2} Test -- True --> LoopBody[Loop Body] LoopBody --> Expr3[Expr3] Expr3 --> Test Test -- False --> Exit(()) </pre>	<pre> /* display first 10 multiples of 2 */ { for(i=1; i <= 10; i++) printf ("2 X %d = %d\n", i, 2*i); } </pre>
----	---	--	--

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
<p>Algorithm SumofN</p> <p>Begin</p> <p>Input n</p> <p>Sum=0</p> <p>While n>0 do</p> <p>sum=sum+n</p> <p>n=n-1</p> <p>Output sum</p> <p>End</p>	<pre> graph TD Start([start]) --> Sum0[Sum=0] Sum0 --> ReadN[/Read n/] ReadN --> Compute[Compute Sum=sum+n] Compute --> Decision{n>0} Decision -- True --> Compute Decision -- False --> Print[/Print value of sum/] Print --> Stop([stop]) </pre>	<pre> #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); } </pre>

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
<p>Algorithm CharCount Begin Count=0 Input ch While ch ≠EOF do Count=count+1 Input ch Output count End</p>	 <pre> graph TD Start([start]) --> Init[Count = 0] Init --> Read[/Read ch/] Read --> Cond{Ch=EOF?} Cond -- False --> Inc[Count = count+1] Inc --> Read Cond -- True --> Print[/Print count/] Print --> Stop([stop]) </pre>	<pre> #include <stdio.h> main() { char ch; int count=0; while((ch=getchar())!=EOF) count++; printf("Total characters =%d", count); } </pre>

Self Activity

1. Execute example 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 example 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.

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 a character, an integer n and display the next n characters.
5. Write a program to accept an integer and check if it is prime or not.
6. Write a program to accept an integer, count number of digits and calculate sum of digits in the number. Example: Number = 1234 Output: Digits = 4, Sum = 10
7. Write a program to accept an integer and reverse the number.
Example: Input: 546, Reverse = 645.

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 to calculate sum of first n terms of the Series $x^1 + x^2 + x^3 + \dots$
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)

Assignment Evaluation

- | | | |
|--------------------------|-------------------|----------------------|
| 0: Not Done [] | 1: Incomplete [] | 2: Late Complete [] |
| 3: Needs Improvement [] | 4: Complete [] | 5: Well Done [] |

Exercise 2**Date:****Objective:**

To demonstrate use of nested loops

Reading

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.

The inner loop has to be completely enclosed in the outer loop. No overlapping of loops is allowed.

Sr. No	Format	Sample Program
1.	<pre> Nested for loop for(exp1; exp2 ; exp3) { for(exp11; exp12 ; exp13) { } } </pre>	<pre> #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 ("%d\t", number); printf ("\n"); } } </pre>
2.	<pre> Nested while loop / do while loop while(condition1) { while(condition2) { } } do { while(condition1) { } } while (condition2); </pre>	<pre> #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; n= n/10; } n=sum; } while(n >9); printf (" %d" , n); } </pre>

Self Activity

- 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
    
```

- 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
    
```

- 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	
—	

Set A . Write C programs for the following problems.

- Write a program to display all prime numbers between ____ and ____.
- 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
    
```

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

```

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

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 n numbers and display the number having the maximum sum of digits.
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)

Assignment Evaluation

0: Not Done []

1: Incomplete []

2: Late Complete []

3: Needs Improvement []

4: Complete []

5: Well Done []

Assignment 3

Date:

Functions (User Defined functions, Library functions and Recursion).

Exercise 1:

To demonstrate menu driven programs and use of standard library functions

Reading

You should read following topics before starting this exercise

1. Use of switch statement to create menus as in exercise3
2. Use of while and do while loops as in

ctype.h : contains function prototypes for performing various operations on characters.

Function	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 : contains function prototypes for performing various mathematical operations on numeric data.

Name	Description
ceil	smallest integer not less than parameter
cos	cosine
cosh	hyperbolic cosine
exp(double x)	exponential function, computes e^x
fabs	absolute value
floor	largest integer not greater than parameter
fmod	floating point remainder
log	natural logarithm
log10	base-10 logarithm
pow(x,y)	compute a value taken to an exponent, x^y
sin	sine
sinh	hyperbolic sine
sqrt	square root
tan	tangent
tanh	hyperbolic tangent

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: ISLOWER "), 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

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. Power
 - ii. Square Root
 - iii. Floor
 - iv. Ceiling
 - v. Exit

Set B. Write C programs for the following problems

1. Accept two fractions (numerator, denominator) and perform the following operations till the user selects Exit.
 - i. Addition
 - ii. Subtraction
 - iii. Multiplication
 - iv. EXIT
2. Accept x and y coordinates of two points and write a menu driven program to perform the following operations till the user selects Exit.
 - iv. Distance between points
 - v. Slope of line between the points.
 - vi. Check whether they lie in the same quadrant.
 - vii. EXIT

Assignment Evaluation

- | | | |
|--------------------------|-------------------|----------------------|
| 0: Not Done [] | 1: Incomplete [] | 2: Late Complete [] |
| 3: Needs Improvement [] | 4: Complete [] | 5: Well Done [] |

Exercise 2:**Date:****Objective:**

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

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 and return only 1 value. In C, every program has a function named main. main in turn calls other 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.

```

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.

```

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;
    }
}

```

Self Activity

1. Type the program given in sample code 1 above and execute the program. 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. 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 check 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.

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:

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

Assignment Evaluation

- | | | |
|--------------------------|-------------------|----------------------|
| 0: Not Done [] | 1: Incomplete [] | 2: Late Complete [] |
| 3: Needs Improvement [] | 4: Complete [] | 5: Well Done [] |

Exercise 3:**Date:****Objective:**

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 when 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.	The recursive definition for factorial is given below: $n! = 1$ if $n = 0$ or 1 $= n * (n-1)!$ 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.	The recursive definition for nCr (no of combinations of r objects out of n objects) is as follows $nCn = 1$ $nC0 = 1$ $nCr = n-1Cr + nCr-1$	<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>

Self Activity

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	—	—		

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 : $\text{gcd}(a,b) = a$ if $b = 0$
 $= \text{gcd}(b, a \text{ mod } b)$ otherwise
3. Write a recursive C function to calculate x^y . (Do not use standard library function)

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:

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

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 4 5 3

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

Assignment Evaluation

0: Not Done [] 1: Incomplete [] 2: Late Complete []
3: Needs Improvement [] 4: Complete [] 5: Well Done []

Assignment 5

Date:

Arrays (1-D and 2-D)

Exercise 1

Objective

To demonstrate use of 1-D arrays and functions.

Reading

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	data-type array_name[size];	int temperature[10]; float pressure[20];
Initialization of array	data-type array_name[]={element1, element2,, element n}; data-type array_name[size]={element-1, element-2,, element-size};	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. Pass the array element by element Pass the entire array to the function	/* Passing the whole array*/ 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

```
#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.

Set A. Write programs to solve the following problems

1. Write a program to accept n numbers and display the array in the reverse order. Write separate functions to accept and display.
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.
Example: Input 1 5 2 1 6 3 8 2 9 15 1 30
Number : 1
Output: 1 occurs 3 times
4. Write a program to accept n numbers and store all prime numbers in an array called prime. Display this array.

Set B. Write programs to solve the following problems

1. 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.
2. Write a program to accept a decimal number and convert it to binary, octal and hexadecimal. Write separate functions.
3. Write a program to find the intersection of the two sets of integers. Store the intersection in another array.
4. 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 26 100

a3 9 10 16 22 25 26 90 100

Assignment Evaluation

- 0: Not Done [] 1: Incomplete [] 2: Late Complete []
3: Needs Improvement [] 4: Complete [] 5: Well Done []

Exercise 2

Objective

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	data-type array_name[size][size];	int mat[10][10]; float sales[4][10];
Initialization of 2-D array	data-type array_name[rows][cols]= { {elements of row 0},{ elements of row 1},.....}; data-type array_name[][cols]={element1, element2,, element size};	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 2-dimensional array - in general, the array element is referred as: array_name[index1][index2] where index1 is the row location of and index2 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.

```
#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.

Set A . Write C programs for the following problems.

1. Write a program to accept a matrix A of size $m \times n$ 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.

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.
 - iv) Check if the matrix is a lower triangular matrix.
 - v) Check if it is an identity matrix.
2. Write a program to accept an $m \times n$ matrix and display an $(m+1) \times (n+1)$ matrix such that the $(m+1)^{\text{th}}$ row contains the sum of all elements of corresponding row and the $(n+1)^{\text{th}}$ 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

Assignment Evaluation

0: Not Done []

1: Incomplete []

2: Late Complete []

3: Needs Improvement []

4: Complete []

5: Well Done []



Savitribai Phule Pune University,Pune

CS113 : Practical course on Problem Solving using
Computer and 'C' programming (CS111) and Database
Management Systems(CS112)

Section B

Database Management Systems

Assignment 1

Date:

Objectives-

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

Reading-

You should read following topics before starting this exercise

1. Designing database 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 as follows:

- a. Character data types:
 - 1 char(n): It is fixed length character string of size n, default value 1 byte if n is not specified.
 - 2 varchar(n): It is variable length character string with maximum size n.
 - 3 text: It is used to store large text data, no need to define a maximum.
- b. Numeric data types:
 - 1 Integer, int, serial: Serial is same as int, only that values are incremented automatically.
 - 2 Number: A real number with P digits, S of them after decimal point.
 - 3 Float: Real number.
- c. Date and time type:
 - 1 Date: Stores date information.
 - 2 Time: Stores time information.
 - 3 Timestamp: Stores a date & time
- d. Boolean and Binary type:
 - 1 Boolean, bool: Stores only 2 values : true or false, 1 or 0, yes or no, y or n, t or f.

There are two constraints applied at the time of creation of table. They can be defined as either of the following :

1. Column level: When data constraint is defined only with respect to one column & hence defined after the column definition, it is called as column level constraint.
Syntax: Create tablename (attribute1 datatype primary key, attribute2 datatype, constraint constraint-name,)
2. 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.
Syntax: Create tablename (attribute1 datatype, attribute2 datatype2,, constraint pkey primary key(attribute1))

Syntax for table creation :

Create tablename (attribute list);

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

Primary key concept : A primary key is made up of one or more columns in a table, that uniquely identify each row in the table. A column or a set of columns defined as a primary key, must conform to the following properties

- a) The column/s cannot have NULL values.
- b) The data held across the column/s MUST be UNIQUE. Syntax:

1. Create tablename (attribute1 datatype primary key , attribute2 datatype ,.....)
2. Create tablename (attribute1 datatype , attribute2 datatype ,....., constraint pkey primary key(attribute1))
3. 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 number(8,2),PRIMARY KEY(order_no,product_no));

Set A

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. Player (player_id int, name varchar(50, Birth_date date ,Birth_place varchar(100))
2. Student (roll_no int,class varchar(20), weight numeric(6,2),height numeric(6,2))
- 3 .Project (project_id int,project_name varchar(20), project_description text,status boolean)
4. Donor (donor_no,donor_name,blood_group,last_date)
5. Movie(movie-no, name, release-year)

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. Teacher (Teacher_no, Tname,qualification,address)

Primary key : Teacher_no

2. Driver (Driver_no,permit_no,Dname,address)

Primary key: Driver_no,permit_no

Assignment Evaluation

0: Not Done []

1: Incomplete []

2: Late Complete []

3: Needs Improvement []

4: Complete []

5: Well Done []

Assignment 2

Date:

Objectives-

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

Reading-

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 referential integrity.

1. Primary key: Designates a column or combination of columns as primary key.
Syntax: PRIMARY KEY (columnname[,columnname])
2. Foreign key: designates a column or grouping of columns as a foreign key with a table constraint.
Syntax: FOREIGN KEY (columnname[,columnname])
3. 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.
Syntax: columnname datatype REFERENCES tablename[(columnname[,columnname])]
4. On delete Cascade: The referential integrity is maintained by automatically removing dependent foreign key values when primary key is deleted.
Syntax: columnname datatype REFERENCES tablename[(columnname)][ON DELETE CASCADE]
5. 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.
Syntax: 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:

- 1 **One-to-one:** A member from an entity set is connected to atmost one member from the other entity set & vice-versa. The key 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.
- 2 **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.
- 3 **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.

4 **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.

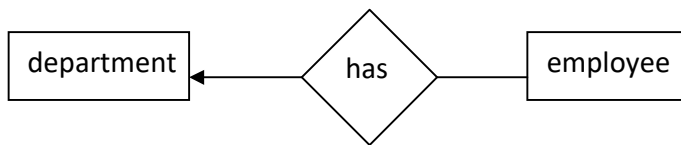
5 **A multivalued, composite attribute:** A composite attribute having multiple values, for each member of the entity set. A new relation is created, which will contain a place holder for each part of the composite attribute and the key attributes of the entity set that contains the composite multivalued attribute.

6 **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 parentchild 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.

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 :

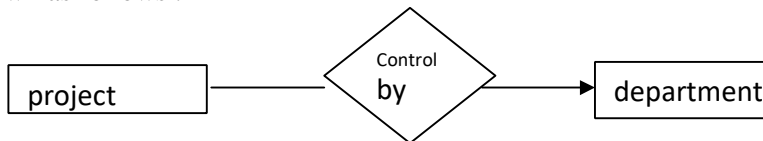


To handle the above relation, while creating the tables, 'deptno' is a foreign key in the employee table. The statement for creating the tables are as follows :

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);

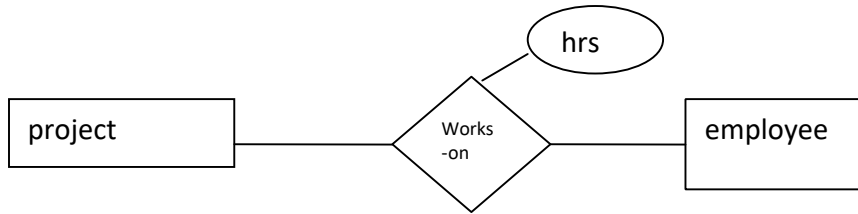
2. Consider the department table created above & another table called project. A project is controlled by exactly one department, but a department can control one or more projects(a m-1 relation). It's pictorially shown as follows :



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))

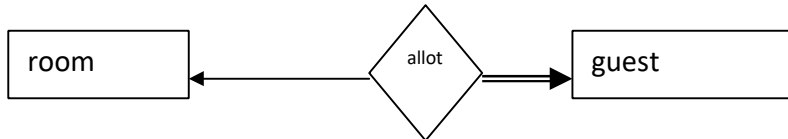
3. Consider the project & employee relations created above. An employee can work in one or more projects, and a project can have one or more employees working in it .(a m-m relation). It's shown pictorially as follows :



To handle the above relationship, we create a ~~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);

Set A

Consider the following entities and their relationships by giving .Create a RDB in 3 NF for the following and create it.

- 1 Hospital(hno int ,name varchar, city varchar)
 Doctor(dno int , dname varchar, city varchar)
 Relationship A many-many relationship between hospital & doctor.
- 2 Patient(pno int ,name varchar ,address text)
 Bed(bedno int , roomno int ,description varchar)
 Relationship A one-one relationship between patient and bed.
- 3..Employee (empno int, name varchar, address text , city varchar, deptname varchar)
 Project (pno int , pname varchar , status)
 Employee and Project are related with many-to-many relationship with attribute - no of days employee worked on that project.
 Constraints: Primary key.

Set B

1. Book (bookno int, name varchar , pubname varchar)

Author (authorno int , author_name varchar)

The relation between book & author is many-to-many with descriptive attribute date_of_publication.

Constraints: Primary Key

Author name & publisher name should not be null.

2. Consider the following Bus transport system .many buses run on one route .Drivers are allotted to the buses shiftwise.

Tables :

Bus (Bus_no int ,capacity int ,depot_name varchar(20))

Route (Route_no int ,source varchar(20),destination varchar(20),no_of_stations int)

Driver(Driver_no int,driver_name varchar(20), licence_no int ,address varchar(20) ,age int ,salary float)

Relationship :

Bus-Route : M-1

Bus-Driver : M-m with descriptive attribute date_of_duty allotted

Constraint :

1.Licence no is unique

2. Bus capacity is not null

3.Shift can be (Morning (1)or Evening (0))

Assignment Evaluation

0: Not Done []

1: Incomplete []

2: Late Complete []

3: Needs Improvement []

4: Complete []

5: Well Done []

Assignment 3

Date:

Objectives-

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

Reading-

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.

- 1 NULL: Specifies that the column can contain null values.
- 2 NOT NULL: Specifies that the column can not contain null values.
- 3 UNIQUE: Forces the column to have unique values.
- 4 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.

Set A

1. Consider the following tables and integrity constraints given and create the tables accordingly:

Machine(Machine_id int Machine_name NOT NULL, Machine_type varchar(10), Machine_price float, Machine_cost float)

- Constraints:
1. Machine_name should be in uppercase.
 2. Machine_type can be ('drilling', 'milling', 'lathe', 'turning', 'grinding').
 3. Machine_price should be greater than zero.

Table level constraint: Machine_cost less than Machine_price.

2. Policy(Policy_no int, Policy_name varchar(20) NOT NULL, Policy_type varchar(10), Policy_sale_date date, Policy_intro_date date)

- Constraints:
1. Policy_name should be in lowercase.
 2. Policy_type can be ('life', 'vehicle', 'accident').

Table level constraint : Policy_sale_date should be greater than Policy_intro_date.

3. 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

Set B

1. Employee(Employee_id int, Employee_name varchar(20) NOT NULL, Employee_desig varchar(10), Employee_sal float, Employee_uid text Unique)

Constraints:

- 1. Employee_name should be in uppercase.
- 2. Employee_desg can be ('Manager', 'staff', 'worker').
- 3. Employee_sal should be greater than zero.

Table level constraint :Employee_uid not equal to Employee_id

2. Consider the following database of bank. A bank maintains the customer details ,account details and loan details .It has the branch information also. Create the tables , in 3NF, as per the information given below.

Account (acct_no int ,acct_type varchar(20),balance int)

Loan(loan_no int,loan_amt int ,no_of_years int)

Branch(branch_no int,branch_name varchar(20),branch_city varchar(20))

Customer(cust_no int ,cust_name varchar(20),cust_address text,customer_city varchar(20))

Relationship : Cutomer-Account : M-M Customer –Loan : M-M Branch –Loan : 1-M Branch –Account : 1-M

- Constraints:
- 1. branch_name should be not null.
 - 2. acct_type should be('saving', 'cuurent')

Assignment Evaluation

0: Not Done []

1: Incomplete []

2: Late Complete []

3: Needs Improvement []

4: Complete []

5: Well Done []

Assignment 4

Date:

Objectives-

To drop a table, alter schema of a table, insert / update / delete records using tables created in previous Assignments. (use simple forms of insert / update / delete statements)

Reading-

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 Drop & Alter DDL statements :

- 1 **Drop:** Deletes an object (table/view/constraint) schema from the database.

Syntax: Drop object-type object-name

- 2 **Alter:** ALTER TABLE command of SQL is used to modify the structure of the table It can be used for

following purposes:

- a) adding new column
- b) modifying existing columns
- c) add an integrity constraint
- d) To redefine a column

Restrictions on the alter table are that, the following tasks cannot be performed with this clause

- a) Change the name of the column
- b) Drop a column
- c) Decrease the size of a column if table data exists

Syntax: a. Alter table tablename Add (new columnname datatype (size), new columnname datatype(size)...);

c.Alter table tablename modify (columnname new datatype(new size),...);

Insert / update / delete DML statements:

The different DML statements with their syntax are given below:(insert / update / delete statements)

1 Insert: The insert statement is used to insert tuples or records into a created table or a relation. We specify a list of comma-separated column values, which must be in the same order as the columns in the table. 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(\).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 along with the values (2nd syntax).

Syntax: INSERT INTO tablename VALUES (list of column values);

INSERT INTO tablename(list of column names) VALUES (list of column values corresponding to the column names);

2. 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.

Syntax: UPDATE tablename

SET columnname = value where condition;

3.Delete: The DELETE statement is used to remove data from tables.

Syntax: DELETE FROM table name where condition;

Self Activity

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;

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',sysdate,'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;

Set A

The Drop & Alter DDL statements

1. Consider the employee table created in assignment 3 and add designation column in the employee table with values restricted to a set of values.
2. 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 2. Add a constraint that the project name should always start with the letter 'R'
4. Consider the relation hospital created in assignment 2. Add a column hbudget of type int , with the constraint that budget of any hospital should always > 50000.

Insert / update / delete DML statements:

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’

Set B

The Drop & Alter DDL statements

1. Create table driver (licence_no, Name, Address) and perform the following queries
 1. Add new column age with constraint that age should be greater than 20 years.
 2. Alter table by modifying driver_name to varchar(50));
 3. Alter table driver ,drop the column age.
 4. Remove the driver table from the database.
2. Create table Game (name, no-of-players, captain_name) and perform the following queries
 1. Add new column game_no with constraint primary key.
 2. Alter table by adding constraint uppercase to captain_name.
 3. Modify table by adding the column game_duration.
 4. Add column game_type with values cricket,hockey,tennis.
 5. Remove game table from the database.

Insert / update / delete DML statements:

1. Consider the following student database .

Tables:

Student(stud_id int , Stud_name varchar(20), S_address varchar(20), status varchar(20))
Teacher(Teacher_id int, Tname varchar(20),T_address varchar(20)) Subject(Subject_id int,
subject_name varchar(20))

Relationship :

Teacher –subject : 1-M

Student-Subject : M-M, with descriptive attributes marks int , grade char.

Constraint :

1. status that should be Pass or Fail.
2. subject name should be not null

Create normalized tables (3NF) for the above and solve the following queries. Write & execute insert/ update / delete statements for following business tasks

- a) Insert 5 records in the student table.
- b) Insert 3 records in the teacher table.
- c) Insert appropriate record in subject and marks table.
- d) Change the marks of 'Ashok' for maths subject to 75.
- e) Change the subject from 'Drawing' to 'Computers'.
- f) Delete the record of teacher 'Sarika'.
- g) Delete all students whose status is fail.

Assignment Evaluation

0: Not Done []

1: Incomplete []

2: Late Complete []

3: Needs Improvement []

4: Complete []

5: Well Done []

Assignment 5

Date:

Objectives-

To query the tables using simple form of select statement Select from table [where order by]
Select from table [where group by <> having <> order by <>]

Reading-

You should read following topics before starting this exercise

1. Creating relations as tables and inserting tuples as records into the relation
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

Different forms of SELECT statement

What do we use SQL for? Well, we use it to select data from the tables located in a database. Immediately, we see two keywords: we need to SELECT information FROM a table. We have the most basic SQL structure:

```
SELECT <column name> FROM <table name>;
```

The select statement : 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.

The basic structure of a Select statement consists of 3 clauses: The select clause corresponds to the projection operation in relational algebra. It is used to list the attributes desired in the query. The from clause corresponds to the Cartesian product operation of RA. It lists the relations to be scanned in the evaluation of the expression. 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.

The other clauses are Order by clause causes the result of the query to appear in a sorted order. Group by clause used to form groups of tuples , of the result . It is used when using aggregate functions.

```
Syntax: select <attribute-list>  
        from <relation-list>  
        [where <condition>  
        [group by <attribute list>  
        [having <condition> ]  
        [order by <attribute list>]]];
```

An SQL aggregate functions performs an operation on a group of rows and returns a single result. You may want retrieve group of item-prices and return total- price. This type of scenario is where you would use a aggregate functions.

The following is the summary of some SQL group function .

- 1 **Sum()**:Gets the sum or total of the values of the specified attribute.

Syntax: Sum(attribute-name)

2 **Count()**: Gives the count of members in the group.

Syntax: Count(attribute); Count(*);

3 **Max()**: Gives the maximum value for an attribute, from a group of members.

Syntax: Max(attribute);

4 **Min()**: Gives the minimum value for an attribute, from a group of members.

Syntax: Min(attribute);

5 **Avg()**: Gives the average value for an attribute, from a group of members.

Syntax: Avg(attribute);

Self Activity

As part of the Set A in exercise 2 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');

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

Set B

As part of the Set B in exercise 3 you have created a Bank Database. Insert sufficient number of records in the relations / tables with appropriate values as suggested by some of the queries. To execute each query type each query into the database prompt and execute

1. Find the number of depositors for each branch.
2. List the branches where the average account balance is more than Rs.1200/-
3. Find all loan numbers that appear in the loan relation with null values for amount.
4. List in alphabetic order all customers who have a loan at the 'city' branch.
5. Find the names of all customers whose street address includes the substring 'Main'.
6. Find the average account balance at city branch.
7. Find the average account balance at each branch.
8. Find the names and loan numbers of all customers who have a loan at the "city branch".
9. Find out the total loan amount at 'Nagar Branch'
10. Find out the total number of customers.

Assignment Evaluation

0: Not Done []

1: Incomplete []

2: Late Complete []

3: Needs Improvement []

4: Complete []

5: Well Done []

Assignment 6

Date:

Objectives-

To query table, using set operations (union, intersect)

Reading-

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 : You can combine multiple queries using the set operators UNION, UNION ALL, INTERSECT and Except. ALL set operators have equal precedence.

- 1 Union: Returns the union of two sets of values, eliminating duplicates.

Syntax: <select query>

Union

<select query>

- 2 Union all: Returns the union of two sets of values, retaining all duplicates.

Syntax: <select query>

Union all

<select query>

- 3 Intersect: Returns the intersection of two sets of values, eliminating duplicates. Syntax: <select query>

intersect

<select query>

- 4 Intersect all: Returns the intersection of two sets of values, retaining duplicates.

Syntax: <select query>

Intersect all

<select query>

- 5 Except: Returns the difference between two set of values, I.e returns all values from set1 , not contained in set2 .eliminates duplicates. Syntax: <select query>

except <select

query>

- 6 Except all: Returns the difference between two set of values, i.e. returns all values from set1, not contained in set2 .Retains all duplicates.

Syntax: <select query>

Except all

<select query>

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.

Self Activity

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;

Set A

Create the following relations :

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.

Set B

1. Create the following tables. (Primary Keys are underlined)

Student(sno,sname,address,class)

Subject(subno,subname)

Student and Subject are related with many-to-many relationship with attribute marks and status.

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

Write the following queries & execute them.

1. List the distinct names of students who have either Electronics, or Statistics or both subjects.
 2. List the names of students who are either passed or failed.
 3. List the students who have “Database” subject and they are not in “TY” class.
 4. List the names of students who are not failed in any subject.
 5. List the names of students not staying at Wagholi.
2. As part of the Set C in exercise 3 you have created a Bank Database. Insert sufficient number of records in the relations / tables with appropriate values as suggested by some of the queries.

To execute each query type each query into the database prompt and execute.

1. Find all customers who have a loan , an account or both.
2. Find all customers who have an account but not loan.
3. Find all customers who have both a loan and an amount.
4. List the customers names having account as well as loan.

Assignment Evaluation

0: Not Done []

1: Incomplete []

2: Late Complete []

3: Needs Improvement []

4: Complete []

5: Well Done []

Assignment 7

Date:

Objectives-

To query tables using nested queries (use of 'Except', exists, not exists, all clauses)

Reading-

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.

1 **Set membership:** The 'in' & 'not in' connectivity tests for set membership & absence of set membership respectively.

2 **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.

3 **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 };
```

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

1 **Exists:** The 'exists' construct returns the value true if the argument subquery is nonempty.

Syntax: select <attribute-list> from <relation-list> where <exists> { sub-query } ;

2 **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)".

Syntax: select <attribute-list> from <relation-list> where <not exists> { sub-query };

Self Activity

1. 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 companyname 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);

2. Consider the table you have prepared as part of self activity of exercise 8, 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.companyname));
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)

Set A

1.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 _____.

2. Consider the table you have prepared as part of Assessment work Set A of exercise 8, 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

Set B

1.Create the following relations:

Movies (M_name,release_year,budget)

Actor (A_name,role,charges,A_address)

Producer (producer_id,name,P_address)

Each actor has acted in one or more movie. Each producer has produced many movies but each movie can be produced by more than one producers. Each movie has one or more actors acting in it, in different roles.

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 actors who have acted in at least one movie, in which 'shahrukh' has acted.
2. List the names of movies with the highest budget.
3. List the names of movies with the second highest budget
4. List the names of actors who have acted in the maximum number of movies.
5. List the names of movies, produced by more than one producer.
6. List the names of actors who are given with the maximum charges for their movie.
7. List the names of producers who produce the same movie as '_____'.
'_____'
8. List the names of actors who do not live in _____ or _____.

2. Consider the table you have prepared as part of Assessment work Set B of exercise 8, and add the following table :

Director (d_no ,d_name ,address)

Consider the given relationship that each director has directed many movies ,and each movie is directed by one and only one director.

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

1. List the names of movies directed by "_____" director.
2. List the names of actors who work in only some movies that "_____" acted on.
3. List the names of the actors who have acted in at least one movie.
4. List the names of actors who have acted in every movie in which _____ has acted.
5. List the names of actors who have acted as a 'villan' in every movie,in which the actor has acted.

Assignment Evaluation

0: Not Done []

1: Incomplete []

2: Late Complete []

3: Needs Improvement []

4: Complete []

5: Well Done []

Assignment 8

Date:

Objectives-

To create views.

Views can be thought of as stored queries, which allow us to create a database object that functions very similarly to a table, but whose contents dynamically reflect the selected rows. Views are very flexible; you may use a view to address common simple queries to a single table, as well as for complicated ones which may span across several tables.

Creating a View : CREATE [OR REPLACE] [TEMP | TEMPORARY] VIEW name [(column_name [, ...])] AS query

CREATE VIEW defines a view of a query. The view has no physical existence, but is dynamically created whenever it is referenced in a query.

CREATE OR REPLACE VIEW replaces an existing view of same name. The new query must use the same column names in the same order and with the same data types, but it may add additional columns to the end of the list.

Schema name (CREATE VIEW myschema.myview) must be specified to create a view in schema other than the current schema. The TEMPORARY or TEMP parameter is specified to create Temporary views, however If any of the tables referenced by the view are temporary, the view is created as a temporary view (whether TEMPORARY is specified or not). Temporary views exist in a special schema, so a schema name cannot be given when creating a temporary view. View names should be distinct.

Name:Name (optionally schema-qualified) of a view to be created.

column_name: An optional list of names to be used for columns of the view. If not given, the column names are deduced from the query.

Query: A SELECT command which will provide the columns and rows of the view.

Example to create a view consisting of all comedy films

```
CREATE VIEW comedies AS
```

```
SELECT *
```

```
FROM films
```

```
WHERE type = 'Comedy';
```

ALTER VIEW statement is used to change the definition of a view.

```
ALTER VIEW name ALTER [ COLUMN ] column SET DEFAULT expression
```

```
ALTER VIEW name ALTER [ COLUMN ] column DROP DEFAULT
```

DROP VIEW statement is used to remove a view

```
DROP VIEW [ IF EXISTS ] name [, ...] [ CASCADE | RESTRICT ]
```

Set A

Using the Bank database

1. Create a view which contains the details of all customers who have applied for a loan more than Rs.100000.
2. Create a view which contains details of all loan applications from the 'Sadashiv peth' branch.
3. Write the following Queries, on the above created views :
 - a. List the details of customers who have applied for a loan of Rs. 500000.
 - b. List the details of loan applications from Sadashiv peth , where loan amount is > Rs 50000.
 - c. List the details of Loan applications, with the same loan amount.

Set B

Using Project-Employee database

1. Create a view over the employee table which contains only employee name and his qualification and it should be sorted on qualification.
2. Create a view containing the project name, project type and start date of the project and should be sorted by start date of the project.
3. Write the following Queries, on the above created views :
 - a. List different qualifications of employees.
 - b. List the name and type of projects started on 1st April 2014.
 - c. List the names of employees who are qualified as Engineers.

Assignment Evaluation

0: Not Done []

1: Incomplete []

2: Late Complete []

3: Needs Improvement []

4: Complete []

5: Well Done []