# **Computer Programming Laboratory (18CPL17/27)**

# **Lab Manual (2020-2021)**

# **Semester I/II**

## **Table of Contents**

[**Computer Programming Laboratory (18CPL17/27)**](#_heading=h.gjdgxs) **1**

[**Lab Manual (2020-2021)**](#_heading=h.gjdgxs) **1**

[**Semester I/II**](#_heading=h.gjdgxs) **1**

[Table of Contents](#_heading=h.30j0zll) [2](#_heading=h.30j0zll)

[Practical #1](#_heading=h.3znysh7) [3](#_heading=h.3znysh7)

[Practical #2](#_heading=h.2et92p0) [5](#_heading=h.2et92p0)

[Practical #3](#_heading=h.tyjcwt) [11](#_heading=h.tyjcwt)

[Practical #4](#_heading=h.3dy6vkm) [15](#_heading=h.3dy6vkm)

[Practical #5](#_heading=h.1t3h5sf) [19](#_heading=h.1t3h5sf)

[Practical #6](#_heading=h.4d34og8) [24](#_heading=h.4d34og8)

[Practical #7](#_heading=h.3rdcrjn) [30](#_heading=h.3rdcrjn)

[Practical #8](#_heading=h.26in1rg) [34](#_heading=h.26in1rg)

[Practical #9](#_heading=h.lnxbz9) [41](#_heading=h.lnxbz9)

[Practical #10](#_heading=h.35nkun2) [45](#_heading=h.35nkun2)

[Practical #11](#_heading=h.44sinio) [54](#_heading=h.44sinio)

[Practical #12](#_heading=h.2jxsxqh) [58](#_heading=h.2jxsxqh)

[PRACTICAL #13](#_heading=h.z337ya) [62](#_heading=h.z337ya)

[PRACTICAL #14](#_heading=h.3j2qqm3) [68](#_heading=h.3j2qqm3)

[Practical #15](#_heading=h.1y810tw) [73](#_heading=h.1y810tw)

[Practical #16](#_heading=h.4i7ojhp) (Extra Program) 75

## 

## Practical #1

**Aim**:

Familiarization with computer hardware and programming environment, concept of naming

the program files, storing, compilation, execution and debugging. Taking any simple C code.

C is a high level language and it needs a compiler to convert it into an executable code. Then, we can execute the program by running the executable code or file.

A compiler is a software that converts a program written in High Level Language to Machine Language. Few examples of compiler are cc, gcc, cpp, java etc.

gcc and cc converts C program to machine language.

cpp converts C++ program to machine language.

java converts java programs to machine language.

**I Execution of C program:**

In order to execute or run a C program the following steps needs to be followed -

1. Edit the program using an editor.
2. Save the program with a proper name.
3. Compile the program.
4. Run the program.

**1. To edit a C program**

To edit a C program we need a text editor. We will be using the editor - **gedit** to edit C programs.

There are two ways to invoke gedit

1. From the Terminal Window
2. From gedit icon
3. **From Terminal Window:**

To edit C program file from terminal window follow the following steps:

1. Open the terminal window either by typing **<ctrl>+<Alt>+t** or by double clicking on the Terminal icon in the Taskbar.
2. .In the Terminal window, at the command prompt - $ (dollar), type ‘gedit <filename>.c’ as shown below

**$ gedit hello.c**

**2. From gedit icon:**

Double click the gedit icon in the taskbar.

If gedit icon is not present in the taskbar, click on **‘Search your Computer**’ icon (The very first icon at the top left corner of the monitor) and type ‘e’ in the search box. Text editor icon is displayed. Drag the icon and place it in the taskbar and double click the gedit icon.

Preferably, invoke gedit from the gedit icon.

The editor opens with an **untitled document.**. Now, type the code for the program. Let us type the very first program of C - The Hello World program. Type the following program in the editor.

/\* Program to print Hello World\*/

#include <stdio.h>.

int main()

{

printf(“Hello World”);

Return 0;

}.

**2. To Save the Program with Proper name and extension:**

After typing the complete program, it must be saved. To save the program press the shortcut key **ctrl+s** or click on the **save** button at the top right corner of the gedit window. If it is not named it has to be given a proper name. Name the file appropriate to the program and the extension must be ‘.c’ (dot followed by lower case c).

Example: hello.c palin.c, sqroot.c, binsrch.c, matmul.c etc.

**Note:** The C compiler is case sensitive, it treats the file as C program file, only if the extension name is “.c” and not “.C”.

**3. To Compile the Program:**

To compile the program, open terminal window and at the prompt, type the following command and press enter key.

**$ cc hello.c**

An output file (executable file) with name **a.out** is created if there are no errors in the program. In case there are errors/warnings, they are displayed on the terminal window.

**To run or Execute the program:**

To run the program, type **./a.out** at the prompt as shown below

**$./a.out**

The program gets executed and the output is displayed on the terminal window.

**II Stages of Compilation:**

**What happens during the compilation process?**

Compiler is a software which converts a program written in high level language such as C into machine language (a language understandable by the computer). The process of conversion involves of four phases, viz.,

1. Pre-processing
2. Compilation
3. Assembly
4. Linking
5. **Pre-processing:**

* The pre-processing phase is the first stage in compilation. During this phase the Preprocessor interprets all preprocessor directives present in the C program. Preprocessor performs the following activities -
* Removal of Comments
* Expansion of Macros
* Expansion of the included files.
* The preprocessor takes the source code (.c file, eg: hello.c) as input and gives the preprocessed code as output. The preprocessed code is stored in the file - **filename.i** (eg: hello.i). The preprocessed code is a text file. We can open and view the contents of the preprocessed code (hello.i).
* By executing below command, We get the preprocessed code -

$**cc -E hello.c -o hello.i**

* Open the file - hello.i and observe the contents of the file. The comments are removed and macros are replaced with corresponding code. Header file has been expanded and included in source file.

**2. Compilation**

* Compilation is the second phase. During this phase the preprocessed code is converted to equivalent assembly code.The input is the preprocessed code (hello.i) and the output is assembly code (hello.s). The assembly code is also a text file and can be opened and viewed using the editor.
* The following command converts the preprocessed code to assembly code

$ cc -S hello.i

* Open the assembly code and observe the contents of the file.This file contains code written in assembly language. This file is taken as input by the assembler and it converts into filename.o output file. This will contain code in machine language.

During the linking phase, the function calls are linked with their definitions. The linker adds additional code to setup the environment for command line arguments. As a result, there will be a significant increase in the size of the file.

**Note: Apart from this please write the LINUX commands studied like:**

## **Basic Linux Commands**

## **1.ls**

## List files and/or directories. If no argument is given, the contents of current directory are shown.

$ ls  
example file1.txt file2.txt file3.txt

If a directory is given as an argument, files and directories in that directory are shown.

$ ls /usr  
bin games include lib lib64 local sbin share src

‘ls -l’ displays a long listing of the files.

$ ls -l  
total 4  
drwxr-xr-x 2 raghu raghu 4096 2012-07-06 12:52 example  
-rw-r--r-- 1 raghu raghu 0 2012-07-06 12:52 file1.txt

-rw-r--r-- 1 raghu raghu 0 2012-07-06 12:52 file2.txt  
-rw-r--r-- 1 raghu raghu 0 2012-07-06 12:52 file3.txt

## **2. cd**

The cd command - change directory - will allow the user to change between file directories. As the name command name suggest, you would use the cd command to circulate between two different directories.

Ex:  
$ pwd  
/home/raghu  
$ cd /usr/share/  
$ pwd  
/usr/share  
$ cd doc  
$ pwd  
/usr/share/doc

$cd .. To exit from current directory

$pwd

/usr/share

## **3. mv – move command**

$ mv source destination

Move files or directories. The 'mv' command works like 'cp' command, except that the original file is removed. But, the mv command can be used to rename the files (or directories).

## **4. pwd-present working directory**

‘pwd’ command prints the absolute path to current working directory.

$ pwd  
/home/raghu

## **5. mkdir**

To create a directory, the ‘mkdir’ command is used.

$ mkdir example  
$ ls -l  
total 4  
drwxr-xr-x 2 raghu raghu 4096 2012-07-06 14:09 example

## **6. rmdir**

 rm files|directories

$ rm file2

'rmdir' command removes any empty directories, but cannot delete a directory if a file is present in it. To use ‘rmdir’ command, you must first remove all the files present in the directory you wish to remove (and possibly directories if any).

## **7 cat**

The 'cat' command is actually a concatenator but can be used to create and view the contents of a file.**.**

**$cat filename**

**Displays the contents of the file**

**$ cat>filename**

Press enter, then you can write something in the file and then to save the file contents press ctrl+d then enter.

## **8. touch**

For creating an empty file, use the touch command.  
$ touch file1 file2 file3

8. copy command - cp

$cp source destination

Copy files and directories. If the source is a file, and the destination (file) name does not exit, then source is copied with new name i.e. with the name provided as the destination.

## 8. rm

Removes files from the directory:

**rm testfile.txt**

## **9. cal**

Displays the calendar of the current month.

|  |  |
| --- | --- |
| $ cal July 2012 Su Mo Tu We Th Fr Sa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 | $ cal 08 1991 August 1991 Su Mo Tu We Th Fr Sa 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 |

## **10. clear**

The clear command does exactly what it says. The clear command clears the screen and wipes the board clean. Using the clear command will take the user back to the start prompt of whatever directory you are currently operating in. To use the clear command simply type $**clear**.

**Questions for viva:**

1. What is a compiler? What is the function of a compiler?
2. Explain intermediate code.
3. Which are the four steps followed to convert a C program to executable code?
4. What is the difference between assembly and machine language?
5. What are the steps followed in preprocessing phase?
6. What happens during the linking phase?
7. Why does the size of the file increase after linking phase?
8. Which are the different LINUX commands and what is their pupose.?

## Practical #2

**Aim**: Develop a program to solve simple computational problems using arithmetic expressions and use of each operator leading to simulation of a commercial calculator. (No built-in math function)

**Problem statement:** Write a C program to demonstrate the functioning of a calculator without using the math library functions.

**Algorithm:**

Input: Two integers(operands) and operator

Output: Result of the operation

1.Start  
2. Read two operands and an operator

3. Check if operator is ‘+’, if yes, then goto step 4 else goto step 5  
4. Compute addition operation:  
 Res = num1 + num2 goto step 18  
5. Check if operator is ‘-’, if yes, then goto step 6 else step 7  
6. Compute subtraction operation  
 Res = num1 - num2 goto step 18  
7. Check if operator is ‘\*’, if yes, then goto step 8 else step 9  
8. Compute multiplication operation  
 Res = num1 \* num2 goto step 18  
9. Check if operator is ‘/’, if yes, then goto step 10 else goto step13  
10. Check if num2 is zero, if yes, then goto step 12 else goto step 11  
11. Compute division operation  
 Res = num1 / num2 and goto step 18  
12. Display “Divide by zero error”, and goto step 19

13. Check if operator is ‘%’, if yes then goto step 14 else goto step 17

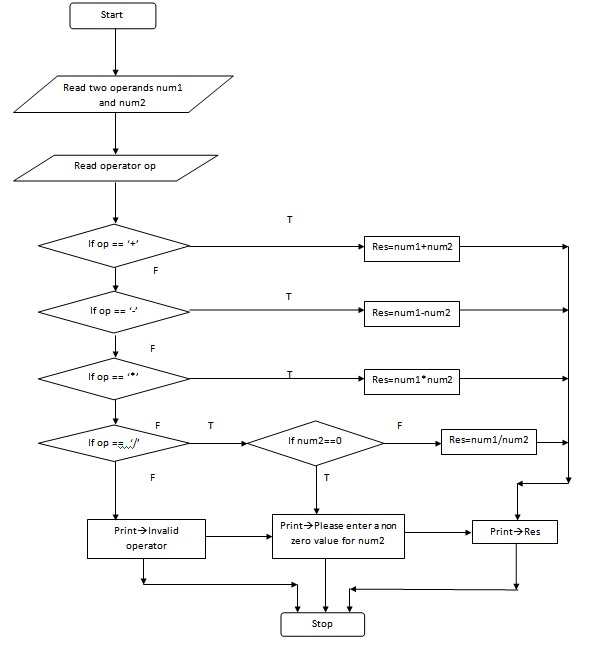
14. Check if num2 is zero, if yes then goto step 16 else goto step 15  
15. Compute division operation  
 Res = num1 % num2 and goto step 18

16. Display “Divide by zero error”. And goto step 19

17. Display “Invalid operator” and goto step 19

18. Display the result  
19. Stop

**Flowchart:**

****

**Program**:

/\* Part - A:   
 \* Program 2:  
 \* Develop a program to solve simple computational problems using arithmetic expressions and  
 \* use of each operator leading to simulation of a commercial calculator. (No built-in math  
 \* function)  
 \*/  
  
 # include<stdio.h>  
   
 int main()  
 {  
 int num1,num2;  
   
 float result;  
 char op;  
   
 printf("Enter two integers :");  
 scanf("%d%d", &num1,&num2);  
   
 printf("Enter the operator \n");  
 printf("+ for addition\n");  
 printf("- for subtraction\n");  
 printf("\* for multiplication\n");  
 printf("/ for division\n");  
 printf("%% for modulus\n"); // % is an escape character of printf's, and \ of compiler.  
   
 scanf(" %c",&op); // Space and %c to accept a character   
   
 if(op=='+')  
 {  
 result=num1+num2;  
 }  
 else if(op=='-')  
 {  
 result=num1-num2;  
 }  
 else if(op=='\*')  
 {  
 result=num1\*num2;  
 }  
 else if(op=='/')  
 {  
 if (num2 == 0)  
 {  
 printf("Retry and enter a non-zero number... \n");  
 return (-1);  
 }  
 result=num1/num2;  
 }  
 else if(op=='%')  
 {  
 if (num2 == 0)  
 {  
 printf("Retry and enter a non-zero number... \n");  
 return (-1);  
 }  
 result=num1%num2;  
 }  
 else  
 {  
 printf("Invalid operator...\n");  
 return 1;  
 }  
   
 printf("Result = %f\n",result);  
 return 0;  
 }  
   
  
**Sample Output:**  
   
 $ cc prog2.c  
 $ ./a.out  
Enter two integers :4  
9  
Enter the operator   
+ for addition  
- for subtraction  
\* for multiplication  
/ for division  
% for modulus  
+  
Result = 13.000000  
  
$ ./a.out  
Enter two integers :12  
45  
Enter the operator   
+ for addition  
- for subtraction  
\* for multiplication  
/ for division  
% for modulus  
-  
Result = -33.000000  
  
$ ./a.out  
Enter two integers :56  
87  
Enter the operator   
+ for addition  
- for subtraction  
\* for multiplication  
/ for division  
% for modulus  
\*  
Result = 4872.000000  
  
$ ./a.out  
Enter two integers :23  
67  
Enter the operator   
+ for addition  
- for subtraction  
\* for multiplication  
/ for division  
% for modulus  
/  
Result = 0.000000  
  
$ ./a.out  
Enter two integers :3 5  
Enter the operator   
+ for addition  
- for subtraction  
\* for multiplication  
/ for division  
% for modulus  
%  
Result = 3.000000  
  
$ cc prog2.c  
$ ./a.out  
Enter two integers :2 0  
Enter the operator   
+ for addition  
- for subtraction  
\* for multiplication  
/ for division  
% for modulus  
/  
Retry and enter a non-zero number...   
  
$ ./a.out  
Enter two integers :2  
0  
Enter the operator   
+ for addition  
- for subtraction  
\* for multiplication  
/ for division  
% for modulus  
%  
Retry and enter a non-zero number...

**Questions for viva:**

1. Why do we need to include a stdio.h file in a C program?
2. What is an operator in C?
3. What are the different types of arithmetic operators?
4. What are the different types of relational operators?
5. What is the difference between / and % operator?
6. Why do we use printf()?
7. Why do we use scanf()?
8. Explain looping structures in C.
9. Explain decision control structures in C.

## Practical #3

**Aim:** Develop a program to compute the roots of a quadratic equation by accepting the coefficients. Print appropriate messages.

**Problem statement:** Design and develop a flowchart or an algorithm that accepts 3 coefficients of a quadratic equation as input and compute all possible roots. Implement a C program for the developed flowchart and execute to find the possible roots for the given set of coefficients and print messages accordingly.

**Algorithm:**

Input: Three non zero coefficients a,b and c of a quadratic equation.

Output: To compute roots for the quadratic equation.

1. Start
2. Read three non zero coefficients a,b and c.
3. Check if a is equal to zero. If true, go to step 4 else go to step 5.
4. Display the equation is linear and go to step 11.
5. Calculate the discriminant as follows:

D= b\*b - 4\* a\*c

1. Check if discriminant is 0. If true, go to step 7, else go to step 8.
2. Compute roots as follows:

Root1 = -b/2\*a

Root2 = root 1

Display: Roots are real and equal. Root1. Go to step 11.

1. Check is discriminant is greater than 0. If true, go to step 9 else go to step 10.
2. Compute roots as follows:

root1= (-b + sqrt(D))/(2\*a)

Root2 = (+b -sqrt(D))/(2\*a)

Display: Roots are real and distinct. Root1 and Root2. Go to step 11.

1. Calculate

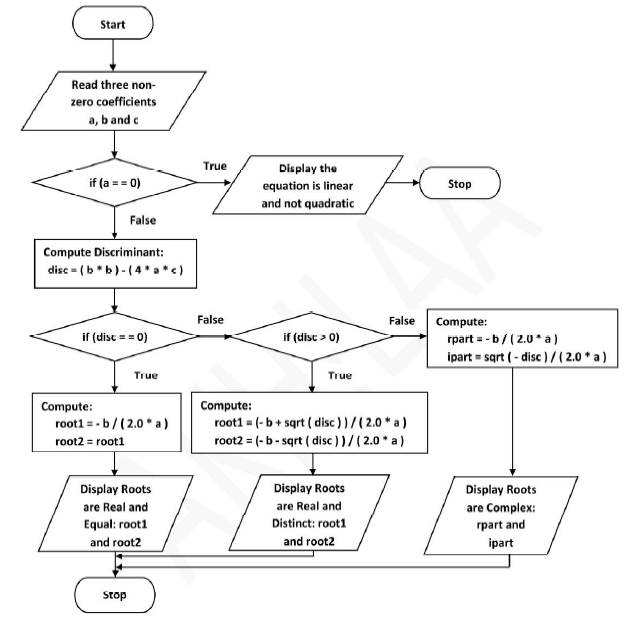
Rpart = -b/2\*a

Ipart = sqrt(-d)/(2\*a)

Display. Roots are imaginary. Rpart and ipart. Go to step 11.

1. Stop

**Flowchart:**

****

**Program code:**

/\* Program 3:  
 \* Develop a program to compute the roots of a quadratic equation by   
 \* accepting the coefficients. Print appropriate messages.  
 \*/  
   
 #include<stdio.h>  
 #include<math.h>  
   
 int main()  
 {  
 float a,b,c,desc,r1,r2,realpart,imgpart;  
   
 printf("Enter the coefficients of a, b and c :");  
 scanf("%f%f%f",&a,&b,&c);  
   
 if(a==0)  
 {  
 printf("Coefficient of a cannot be zero....\n");  
 printf("Please try again....\n");  
 return 1;  
 }  
   
 desc=(b\*b)-(4.0\*a\*c);  
   
 if(desc==0)  
 {  
 printf("The roots are real and equal\n");  
 r1=r2=(-b)/(2.0\*a);  
 printf("The two roots are r1=r2=%f\n",r1);  
 }  
 else if(desc>0)  
 {  
 printf("The roots are real and distinct\n");  
 r1=(-b+sqrt(desc))/(2.0\*a);  
 r2=(-b-sqrt(desc))/(2.0\*a);  
 printf("The roots are r1=%f and r2=%f\n",r1,r2);  
 }  
 else  
 {  
 printf("The roots are imaginary\n");  
 realpart=(-b)/(2.0\*a);  
 imgpart=sqrt(-desc)/(2.0\*a);  
 printf("The roots are r1=%f + i %f\n",realpart,imgpart);  
 printf("r2=%f - i %f\n",realpart,imgpart);  
 }  
 return 0;  
 }

**Questions for viva:**

1. What are library files? How are they called in a C program?
2. What are format strings? Explain with example.
3. How can we find the square root of a given number without using sqrt()?
4. What is a preprocessor? Which preprocessor is used in this program?
5. Why do we use \n ?
6. What is the return type of main function? Can we use void()? If yes, explain what changes need to be made to the program.

## Practical #4

**Aim**: Develop a program to find the reverse of a positive integer and check for palindrome or not. Display appropriate messages.

**Problem statement:** Write a C program to calculate the reverse of a positive integer. Also, need to check whether it is palindrome or not.

Enter a number: 2018

The reverse of the number is 8102. Not a palindrome.

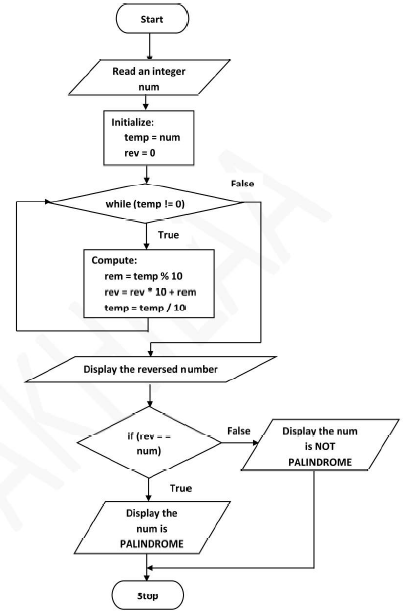
**Algorithm:**

Input: Positive integer

Output: The reverse of the number. Palindrome or not palindrome.

1. Start
2. Read integer num.
3. Find the reverse of num.
4. To get quotient, num/10. To get remainder, num%10.
5. Check if quotient is not zero. If true, go to 3a. If false, go to step 4.
6. Display the reverse number.
7. Check if reverse is equal to num. If true, display palindrome. If false, display not a palindrome.
8. Stop.

**Flowchart:**

****

**Program**:

/\* Program 4  
 \* Develop a program to find the reverse of a positive integer and check   
 \* for palindrome or not. Display appropriate messages.  
 \*/  
   
 #include <stdio.h>  
   
 int main()  
 {  
 int num,temp,rev,rem;  
   
 // Accept the input from the user  
 printf("Enter a positive integer :");  
 scanf("%d", &num);  
   
 // Initialise the variables  
 temp = num;  
 rev=0;  
   
 // Reverse the given number  
 while (temp != 0)  
 {  
 rem = temp % 10;  
 rev = rev \* 10 + rem;  
 temp = temp / 10;  
 }  
   
 // Display the reversed number  
 printf("The reversed number is %d\n",rev);  
   
 // The number is a palindrome if given number is same as reversed number.  
 // Check if the given number is same as reversed number  
 if(num == rev)  
 {  
 printf("The number %d is a palindrome\n", num);  
 }  
 else  
 {  
 printf("The number %d is not a palindrome\n", num);  
 }  
   
 return (0);  
 }  
   
 **Sample Outputs:**

1. $ cc prog4.c  
 $ ./a.out  
 Enter a positive integer :1234  
 The reversed number is 4321  
 The number 1234 is not a palindrome  
   
 2. $./a.out  
 Enter a positive integer :1221  
 The reversed number is 1221  
 The number 1221 is a palindrome  
   
 3. $./a.out  
 Enter a positive integer :121121  
 The reversed number is 121121  
 The number 121121 is a palindrome  
   
 4. $./a.out  
 Enter a positive integer :33244233  
 The reversed number is 33244233  
 The number 33244233 is a palindrome  
  
 5. $ ./a.out  
 Enter a positive integer :5467382  
 The reversed number is 2837645  
 The number 5467382 is not a palindrome

**Questions for viva:**

1. What do you mean by pre-processor directive?
2. What is a header file? Give examples.
3. Why do we use main()?
4. What are format specifiers?
5. What are expressions?
6. What do you mean by precedence?
7. What do you mean by associativity?
8. Difference between sqrt(), sqrtf() and sqrtl()?

## Practical #5

**Aim**: An electricity board charges the following rates for the use of electricity: for the first 200 units 80 paise per unit: for the next 100 units 90 paise per unit: beyond 300 units Rs 1 per unit. All users are charged a minimum of Rs. 100 as meter charge. If the total amount is more than Rs 400, then an additional surcharge of 15% of total amount is charged. Write a program to read the name of the user, number of units consumed and print out the charges.

**Problem statement**: Write a C program to demonstrate the calculations followed while computing electricity bill.

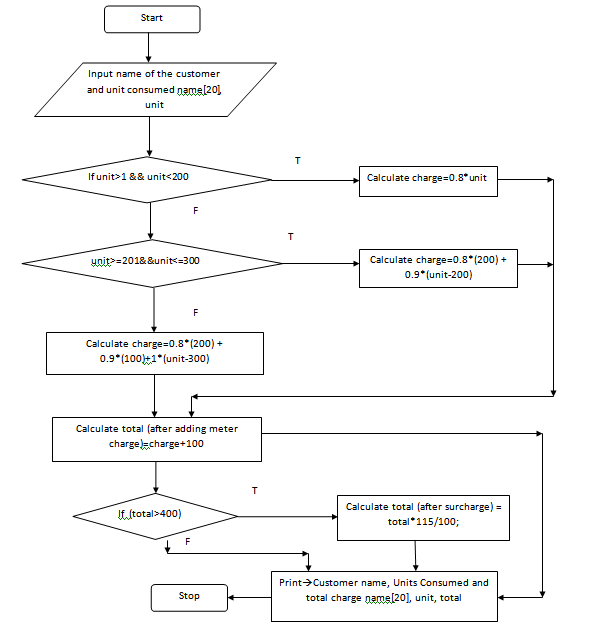
**Algorithm**:

Input: Customer name and unit consumed

Output: Customer name, units consumed and total amount to be paid

1. Start  
2. Read the name of customer and the unit consumed by the customer  
3. Check if the unit consumed is greater than 1 and less than 200,if true goto step 4 else goto step 5  
4. Compute: Charge=0.8\*unit consumed   
5. Check if unit is greater than or equal 201 and less than 300,if true goto step 6 else goto step 7  
6. Compute:Charge=0.8\*(200)+0.9\*(unit 200)  
7. Compute:Charge=0.8\*200+0.9\*(100)+1\*(unit-300)  
8. After calculating charge,find total:  
 Total=charge+100  
9. Check if total is greater than 400,if true goto step 10 else goto step 11  
10. Compute total(after surcharge)=total \*115/100 and proceed to step 11  
11. Display the customer name,Units Consumed and total charge per unit  
12. Stop

**Flowchart:**

****

**Program Code:**

/\* Program 5  
 \* An electricity board charges the following rates for the use of electricity:   
 \* for the first 200 units 80 paise per unit:   
 \* for the next 100 units 90 paise per unit:   
 \* beyond 300 units Rs 1 per unit.   
 \* All users are charged a minimum of Rs. 100 as meter charge.   
 \* If the total amount is more than Rs 400, then an additional surcharge   
 \* of 15% of total amount is charged. Write a program to read the name  
 \* of the user, number of units consumed and print out the charges.  
 \*/  
   
#include <stdio.h>  
   
int main()  
 {  
 float unit,total,charge;  
 char name[20];  
   
 // Accept Customer's name  
 printf("Enter the name : ");  
 gets(name);  
   
 // Accept No. of units consumed  
 printf("Enter the units : ");  
 scanf("%f",&unit);  
  
 charge=0;  
   
 if(unit>=1&&unit<=200) // Charge 80 paise per unit for the first 200 units  
 {  
 charge=0.8 \* unit;  
 }  
 else if(unit>=201&&unit<=300) // Charge 80 paise per unit for the first 200 units and   
 { // 90 paise per unit for the next 100 units   
 charge = 0.8\*(200) + 0.9\*(unit-200);  
 }  
 else if (unit > 300) //Charge 80 paise per unit for the first 200 units,   
 { // 90 paise per unit for the next 100 units and  
 // one rupee per unit for all units above 300  
 charge = 0.8\*(200) + 0.9\*(100) + 1\*(unit-300);   
   
 }  
 total=charge+100; // Add Meter charge of Rs. 100  
 if(total>400)   
 {  
 total = total + (0.15\*total); // Add additional surcharge of 15 percent of total amount  
 }  
   
 // Display the electricity bill  
 printf("\n\nELECTRICITY BILL\n");  
 printf("----------------\n");  
   
 printf("\nName : %s\n",name);  
 printf("No. of units: %.2f\n",unit);  
 printf("Total Bill Amount: Rs. %.2f\n",total);  
 printf("----------------\n");  
   
 return 0;  
}  
  
**Sample Outputs:**  
1.  
$ cc prog5.c  
$ ./a.out  
Enter the name : Anitha  
Enter the units : 0  
  
  
ELECTRICITY BILL  
----------------  
  
Name : Anitha  
No. of units: 0.00  
Total Bill Amount: Rs. 100.00  
----------------  
  
2.  
$ ./a.out  
Enter the name : Harish  
Enter the units : 167  
  
  
ELECTRICITY BILL  
----------------  
  
Name : Harish  
No. of units: 167.00  
Total Bill Amount: Rs. 233.60  
----------------  
  
3.  
$ ./a.out  
Enter the name : Anuradha  
Enter the units : 598  
  
  
ELECTRICITY BILL  
----------------  
  
Name : Anuradha  
No. of units: 598.00  
Total Bill Amount: Rs. 745.20  
----------------  
4.  
$ ./a.out  
Enter the name : Rajeev Gupta  
Enter the units : 876  
  
ELECTRICITY BILL  
----------------  
  
Name : Rajeev Gupta  
No. of units: 876.00  
Total Bill Amount: Rs. 1064.90  
----------------  
**Questions for viva:**

1. What is the difference between getc(), getch() and gets()?
2. What is the difference between printf() and scanf()?
3. What are looping structures in C? Which looping structure have you used in your C program? Justify.
4. What are control structures in C? Give some examples.
5. Suppose you want to limit the input by user upto 2 decimal places. Which format specifier will you use?
6. When do we use && operator?
7. What are relational operators? Give some examples.

## Practical #6

**Aim:** Introduce 1D Array manipulation and implement Binary search.

**Problem statement:** Write a C program to demonstrate the performance of binary search.

**Algorithm**:

Input: List of unsorted elements and element to be searched.

Output: Sorted list and element is present or not.

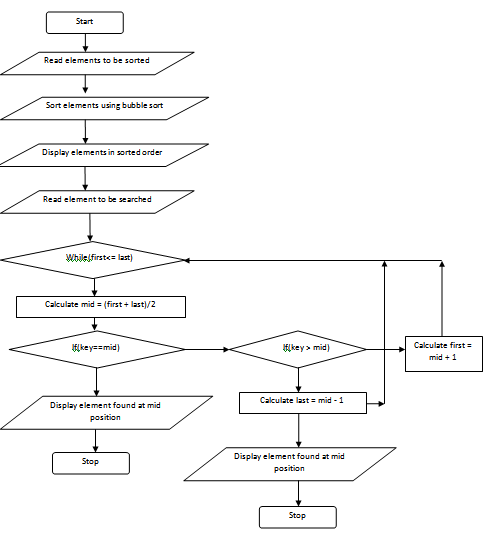
1. Start
2. Read size of the array and list of elements which are not sorted.
3. Use swap function in a loop to sort the given array.
4. Display the sorted list of elements.
5. Read the element to be searched. key.
6. Check if first is not equal to last. If true, find the middle element.
7. A. Check if the key is equal to the middle element. If true, element is found at n/2 index and exit loop. Go to step 8.

B. Else, check if the key is greater than middle element. If true, change first to mid. Go to step 6.

C. Else, check if the key is smaller than middle element. If true, change last to mid. Go to step 6.

8. Stop.

**Flowchart:**

****

**Program:**

/\* Program 6:  
 \* Introduce 1 D Array manipulation and implement   
 \* binary search  
 \*/  
   
 #include <stdio.h>  
 int main()  
 {  
 // Define variables  
 int a[20];  
 int n,i,j,temp,key;  
 int first,mid,last;  
   
 // Accept the size of the array  
 printf("Enter the size of the array :");  
 scanf("%d",&n);  
   
 // Accept elements into the array  
 printf("Enter %d elements :",n);  
 for(i=0;i<n;i++)  
 {  
 scanf("%d",&a[i]);  
 }  
   
 // Print the elements of the array before sorting  
 printf("The elements of the array before sorting is ----\n");  
 for(i=0;i<n;i++)  
 {  
 printf("%d\t",a[i]);  
 }  
   
 /\* Sort the array as the data  
 must be sorted for binary search  
 \*/  
   
 for(i=0;i<n-1;i++)  
 {  
 for(j=0;j<n-1-i;j++)  
 {  
 if(a[j]>a[j+1])  
 {  
 temp=a[j];  
 a[j]=a[j+1];  
 a[j+1]=temp;  
 }  
 }  
 }  
   
 // Display the sorted elements of the array  
 printf("\n\nThe sorted array is ---\n");  
 for(i=0;i<n;i++)  
 {  
 printf("%d\t",a[i]);  
 }  
   
 // Accept the element to be searched  
 printf("\n\nEnter the element to be searched :");  
 scanf("%d",&key);  
   
 // search for the element in the sorted array   
 first=0;  
 last=n-1;  
   
 while(first <= last)  
 {  
 mid=(first+last)/2;  
 if(key==a[mid])  
 {  
 printf("\nThe element %d is found at location %d\n",key,mid+1);  
 return (0);  
 }  
 else if (key < a[mid])  
 {  
 last = mid-1;  
 }  
 else  
 {  
 first = mid+1;  
   
 }  
 }  
   
 printf("\nThe element %d is not found in the array\n",key);  
 return (1);  
 }  
   
**Sample Output:**  
Sample Output 1: Search for the first element in the array  
  
$ ./a.out  
Enter the size of the array :5  
Enter 5 elements :87  
0  
4  
2  
3  
The elements of the array before sorting is ----  
87 0 4 2 3   
  
The sorted array is ---  
0 2 3 4 87   
  
Enter the element to be searched :0  
  
The element 0 is found at location 1  
  
Sample Output 2: Search for the middle element in the array  
  
$ cc prog6.c  
$ ./a.out  
Enter the size of the array :5  
Enter 5 elements :2  
3  
4  
87  
0  
The elements of the array before sorting is ----  
2 3 4 87 0   
  
The sorted array is ---  
0 2 3 4 87   
  
Enter the element to be searched :4  
  
The element 4 is found at location 4  
  
Sample Output 3: Search for middle element in the array  
  
$ ./a.out  
Enter the size of the array :5  
Enter 5 elements :2  
0  
4  
3  
87  
The elements of the array before sorting is ----  
2 0 4 3 87   
  
The sorted array is ---  
0 2 3 4 87   
  
Enter the element to be searched :3  
  
The element 3 is found at location 3  
  
  
Sample Output 4: Search for the last element in the list  
  
$ ./a.out  
Enter the size of the array :5  
Enter 5 elements :3  
2  
4  
0  
87  
The elements of the array before sorting is ----  
3 2 4 0 87   
  
The sorted array is ---  
0 2 3 4 87   
  
Enter the element to be searched :87  
  
The element 87 is found at location 5  
\*/

**Questions for viva:**

1. What is binary search?
2. Why do we need to perform sorting?
3. What is an array? Give example.
4. Can an array store both integer and float variables?

## 

## 

## Practical #7

**Aim:** Implement using functions to check whether the given number is prime and display appropriate messages. (No built-in math function)

**Problem statement:** Write a C program to check whether the given number is prime number or not. Do not use math library functions.

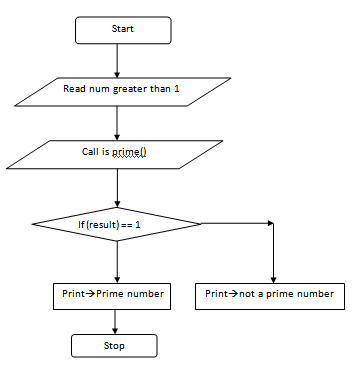
**Algorithm:**

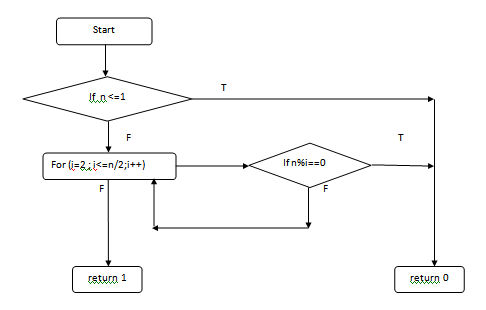
Input : Any integer value

Output: The entered values is prime or not.

1. Start.
2. Read integer number from user.
3. Check whether the number is less than or equal to 1. If true, return 0. If false, go to step 4.
4. Using for loop, check whether the number is divisible or not.
5. If divisible, terminate loop and return 1. If not divisible, terminate loop and return 0.
6. If 1, display number is prime. If 0, display number is not prime.
7. Stop.

**Flowchart:**



****

**Program:**

/\* Program 7  
 \* Implement using functions to check whether the given  
 \* number is prime and display appropriate messages.  
 \* (No build-in math function)  
 \*/  
   
#include<stdio.h>  
  
// Declare the prototype of the user defined function  
int isprime(int);  
  
int main()  
{  
 int num,result;  
   
 // Accept the number  
 printf("Enter a natural number greater than one :");  
 scanf("%d",&num);  
  
 // Call the function isprime() to check whether  
 // the given number is prime or not  
 result = isprime(num);  
   
 if(result == 1) // one means true  
 {  
 printf("The number %d is a prime number\n",num);  
 }  
 else  
 {  
 printf("The number %d is not a prime number\n",num);  
 }  
   
 return (0);  
} // End of main() function  
  
// Definition of isprime() function starts   
int isprime(int n)  
{  
 int i;  
 if (n<=1)  
 {  
 return 0;  
 }  
 for(i=2;i<n/2;i++)  
 {  
 if(n%i==0)  
 {  
 return 0;  
 }  
 }  
 return 1;  
} // End of definition of isprime() function  
  
**Sample Output:**

cc prog7.c  
$ ./a.out  
Enter a positive number :0  
The number 0 is not a prime number  
  
$ ./a.out  
Enter a positive number :1  
The number 1 is not a prime number  
  
$ ./a.out  
Enter a natural number greater than one :-20  
The number -20 is not a prime number  
  
$ ./a.out  
Enter a positive number :87  
The number 87 is not a prime number  
  
$ ./a.out  
Enter a positive number :343  
The number 343 is not a prime number  
  
$ ./a.out  
Enter a positive number :987  
The number 987 is not a prime number  
  
$ ./a.out  
Enter a positive number :67  
The number 67 is a prime number  
  
$ ./a.out  
Enter a positive number :1223  
The number 1223 is a prime number

**Questions for viva:**

1. What is a prime number?
2. Which inbuilt function can be used to check if the number is prime or not?
3. With reference to the program, what will happen if the user gives a negative number as input?
4. What is a function?
5. What are the different types of functions?
6. What is user defined function?

## Practical #8

**Aim**: Develop a program to introduce 2D Array manipulation and implement Matrix multiplication and ensure the rules of multiplication are checked.

**Problem statement:** Write a C program to input matrices A and B. Check for the number of columns in first matrix is equal to number of rows in second matrix. If true, perform matrix multiplication.

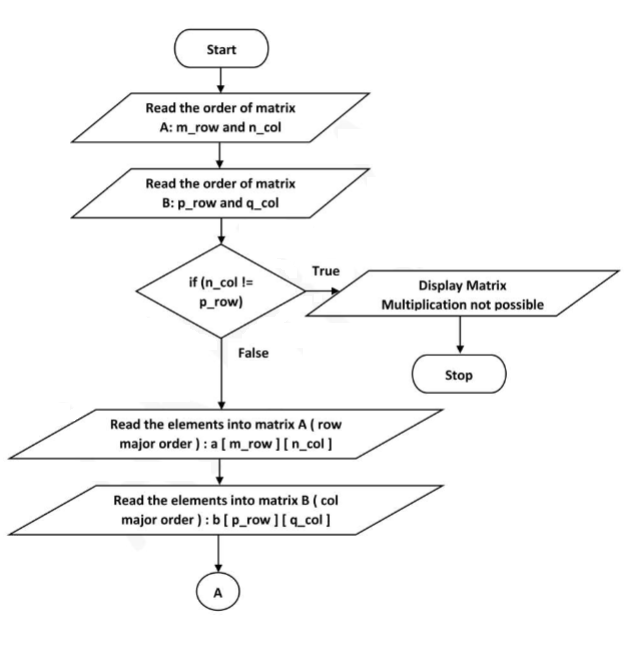
**Algorithm:**

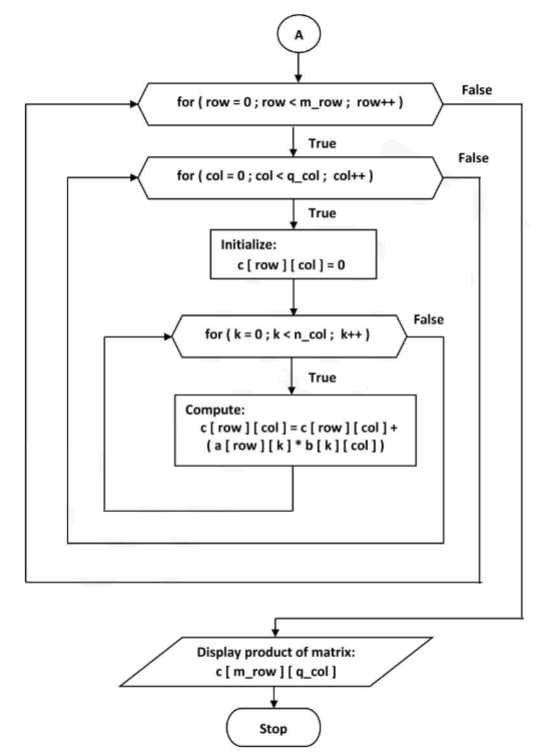
Input: Two matrices A and B.

Output: Result of A x B.

1. Start
2. Input 2 matrices A and B and store them in 2D arrays. Display A and B.
3. Check multiplication condition where the number of columns in A is equal to the number of rows in B. If true, go to step 4. If false, display that matrix multiplication is not possible.
4. Perform matrix multiplication. Display resultant matrix.
5. Exit.

**FLOWCHART**

****

****

**Program:**

/\* Program 8  
 \* Develop a program to introduce 2D Array manipulation and implement   
 \* Matrix multiplication and ensure the rules of multiplication are checked.  
 \*/  
   
   
#include<stdio.h>  
  
int main()  
{  
 int a[10][10],b[10][10],c[10][10];  
 int m,n,p,q;  
 int i,j,k;  
   
 // Input the order of Matrix A - m x n  
 printf("Enter the order of matrix A :");  
 scanf("%d%d",&m,&n);  
   
 // Input the order of Matrix B - p x q  
 printf("Enter the order of matrix B:");  
 scanf("%d%d",&p,&q);  
   
 /\* For multiplication of two matrices, the number of columns in the first  
 matrix should be equal to the number of rows in the second matrix \*/  
 if(n!=p)  
 {  
 printf("Number of columns of Matrix A is not equal to number of rows of matrix B\n");  
 printf("Multiplication of matrices not possible....\n");  
 return (-1);  
 }  
   
 // Input the elements into Matrix A  
 printf("\nEnter %d elements into matrix A : ", m\*n);  
 for(i=0;i<m;i++)  
 for(j=0;j<n;j++)  
 scanf("%d",&a[i][j]);  
   
 // Display matrix A in matrix format  
 printf("\nThe matrix A is ---\n");  
 for(i=0;i<m;i++)  
 {  
 for(j=0;j<n;j++)  
 {  
 printf("%d\t",a[i][j]);  
 }  
 printf("\n");  
 }   
   
 // Input the elements into Matrix B  
 printf("\nEnter %d elements into matrix B : ", p\*q);  
 for (i=0;i<p;i++)  
 for (j=0;j<q;j++)   
 scanf("%d",&b[i][j]);  
   
 // Display Matrix B in matrix format   
 printf("\nThe matrix B is ---\n");  
 for(i=0;i<p;i++)  
 {  
 for(j=0;j<q;j++)  
 {  
 printf("%d\t",b[i][j]);  
 }  
 printf("\n");  
 }  
   
 // Compute (Matrix A) X (Matrix B)  
 for(i=0;i<m;i++)  
 {  
 for(j=0;j<q;j++)  
 {  
 c[i][j] = 0;  
 for(k=0;k<n;k++)  
 {  
 c[i][j] = c[i][j] + (a[i][k] \* b[k][j]);  
 }  
 }  
 }  
   
   
 // Display product matrix - Matrix C  
 printf("\nThe product matrix is ---\n\n");  
 for(i=0;i<m;i++)  
 {  
 for(j=0;j<q;j++)  
 {  
 printf("%d\t",c[i][j]);  
 }  
 printf("\n");  
 }  
   
 return 0;  
}  
  
  
**Sample output:**  
1.  
$cc prog8.c  
$./a.out  
Enter the order of matrix A :2  
2  
Enter the order of matrix B:2  
2  
Enter 4 elements into matrix A : 1  
2  
3  
4  
The matrix A is ---  
1 2   
3 4   
Enter 4 elements into matrix B : 1  
3  
2  
4  
The matrix B is ---  
1 3   
2 4   
The product matrix is ---  
  
5 11   
11 25   
  
2.  
$./a.out  
Enter the order of matrix A :3  
2  
Enter the order of matrix B:2  
3  
  
Enter 6 elements into matrix A : 6  
5  
4  
3  
2  
1  
  
The matrix A is ---  
6 5   
4 3   
2 1   
  
Enter 6 elements into matrix B : 1  
3  
5  
2  
4  
6  
  
The matrix B is ---  
1 3 5   
2 4 6   
  
The product matrix is ---  
  
16 38 60   
10 24 38   
4 10 16

**Questions for viva:**

1. What is a array? What are the different types of arrays?
2. Consider a 5 x 5 matrix. What will be the index to fetch the element at the 3rd row and 3rd column?
3. What is increment operator?
4. What is decrement operator?
5. What is right to left associativity?
6. What is left to right associativity?

## Practical #9

**AIM**

To write a program that implements the Taylor series approximation given by Sin(x) = x - (x3/3!) + (X5/5!) - (x7/7!) + ….

**PROBLEM STATEMENT**

Draw the flowchart and write a C program to compute Sin(x) using Taylor series approximation given by Sin(x) = x - (x3/3!) + (X5/5!) - (x7/7!) + …. Compare your result with the built-in library function. Print both the results with appropriate messages

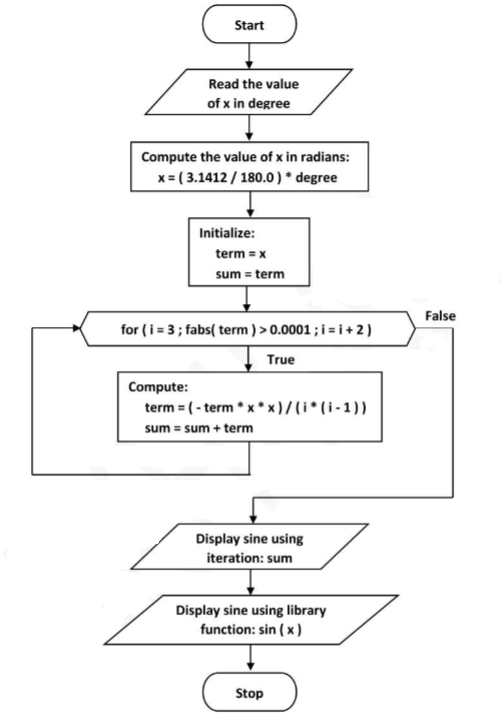
**ALGORITHM**

Input: The value of x in degree

Output: To compute Sin(x) using Taylor Series.

1. Start
2. Read the value of Xin defree
3. Compute the value of x in radians: x = (3.1412/180.0) \* degree
4. Initialise:
   1. Term = x
   2. Sum = term
   3. i=3
5. Check if the absolute value term is greater than 0.0001. If true goto Step 6 else goto Step 7.
6. Compute:
   1. Term = ( - term \* x \* x ) / ( i \* (i - 1 ) )
   2. Sum = sum + term
   3. I = i + 2
7. Display sine using iteration : sum
8. Display sine using library function: sin(x)
9. Stop

**FLOWCHART**



**PROGRAM**

/\* Program 9:  
 \* Develop a Program to compute Sin(x) using Taylor series approximation.  
 \* Compare your result with the built- in Library function. Print both   
 \* the results with appropriate messages.  
 \*  
 \* Taylor Series:  
 \* Sin(x) = x - x3/3! + x5/5! - x7/7! + x9/9! - ......  
 \*/  
   
 #include<stdio.h>  
 #include<math.h>  
   
 #define PI 3.142857  
   
 int main()  
 {  
 float x,degree,nume,deno,sum,term;  
 int i;  
   
 /\* Accept value of x in degree \*/  
 printf("Enter degree:");  
 scanf("%f",&degree);  
   
 // Convert degree into radians  
 x=degree\*(PI/180.0);   
   
 // Initialize values of sum, nume, deno and i variables  
 sum=0;  
 nume=x;  
 deno=1.0;  
 i=1;  
   
 do  
 {  
 // compute term  
 term=nume/deno;  
 // Add term value to sum  
 sum=sum+term;  
 i=i+2;  
 // compute next numerator and denominator values  
 nume=-nume\*x\*x;  
 deno=deno\*i\*(i-1);  
 // printf("Term=%f\n",term);  
 } while (fabs(term) >= 0.00001);  
   
 printf("Computed value of Sin(%f)=%f\n",degree,sum);  
   
 printf("Value from library function is sin(%f) = %f\n",degree,sin(x));  
   
 return 0;  
 }  
   
   
/\*

**Sample Output:**  
1.  
$cc prog9.c -lm  
$./a.out  
Enter degree:60  
Computed value of Sin(60.000000)=0.866236  
Value from library function is sin(60.000000) = 0.866236  
  
2.  
$ ./a.out  
Enter degree:30  
Computed value of Sin(30.000000)=0.500182  
Value from library function is sin(30.000000) = 0.500182  
   
3.  
$ ./a.out  
Enter degree:90  
Computed value of Sin(90.000000)=1.000000  
Value from library function is sin(90.000000) = 1.000000  
  
4.  
$ ./a.out  
Enter degree:45  
Computed value of Sin(45.000000)=0.707330  
Value from library function is sin(45.000000) = 0.707330  
  
5.  
$ ./a.out  
Enter degree:0  
Computed value of Sin(0.000000)=0.000000  
Value from library function is sin(0.000000) = 0.000000  
\*/

**QUESTIONS FOR VIVA**

1. What is the difference between sin(), sinf(). sinl()?

## Practical #10

**Aim:** Write functions to implement string operations such as compare, concatenate, string length. Convince the parameter passing techniques.

**Problem Statement:** Write a C program to demonstrate the working of string functions given in string library file.

**Algorithm:**

10.a.String length:  
1. Start  
2. Read the string given by the user  
3. To find the length of string, call the function str\_length (str)  
4. In str\_length (str): count the characters in the string one by one until it encounters null character then goto step 5  
5. Display the length of string.

10.b String Compare:  
1. Start  
2. Read two strings str1 and str2 given by the user  
3. To find comparison between two strings goto step 4  
4. Call the function str\_compare(str1,str2)  
 a. Compare ASCII value of a character in string1 with string2,if both value equals then

goto step 5  
 b. Check if ASCII value of character in string1 is greater than string2 character then

goto step 6  
 c. Check if ASCII value of character in string1 is lesser than string2 character then goto

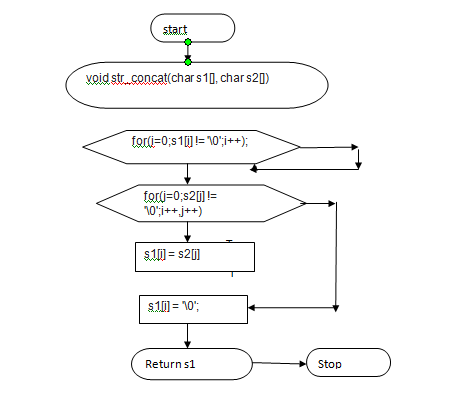
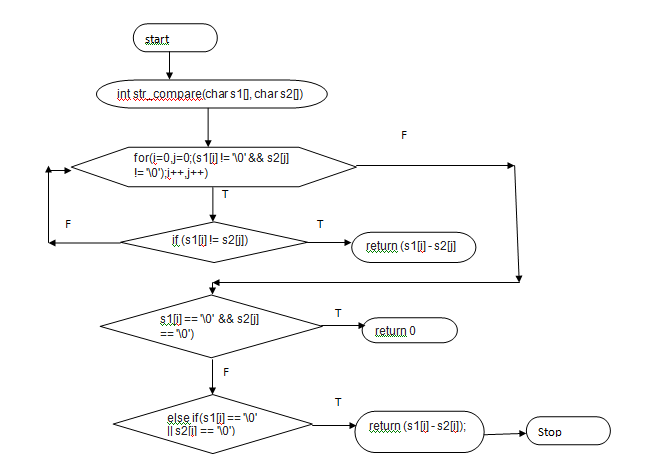
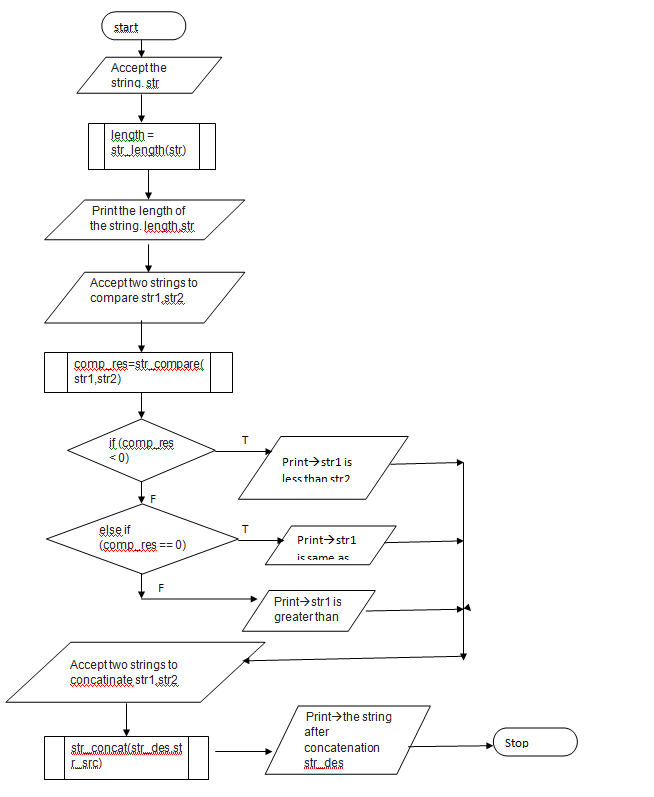
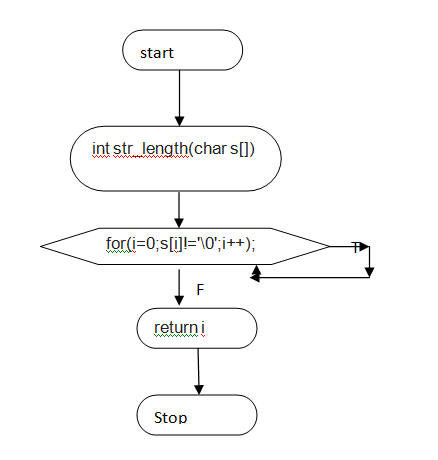
step 7  
5. Display both strings are same  
6. Display String1 is greater than String2  
7. Display String1 is lesser than String2  
8. Stop

10. c. String Concatenation  
1. Start  
2. Read two strings str1, str2 from user  
3. Call the concatenation function str\_concat (str1, str2)

a. Find the last character in string1 by checking the null character,then copy each

character in string2 to string1  
 b. After concatenation of string2 to string1, append null character at the end of string1.  
4. Display the concatenated string  
5. Stop

**FLOWCHART**

****

**PROGRAM**

/\* Program 10  
 \* Write functions to implement string operations such as compare,   
 \* concatenate, string length. Convince the parameter passing techniques.  
 \*/  
   
 #include <stdio.h>  
   
   
 // Prototypes for the user defined functions  
 int str\_length(char []);   
 int str\_compare(char [], char []);  
 void str\_concat(char [], char []);  
   
 int main()  
 {  
 // Declare the variables  
 char str[50];  
 char str1[50], str2[50];  
 char str\_des[100], str\_src[50];  
   
 int length, comp\_res;  
   
 // Accept the string from the user to find the length   
 printf("\nEnter a string :");  
 scanf("%s",str);  
   
 // Invoke the function to find the length of the string  
 length = str\_length(str);  
   
 // Print the length of the string  
 printf("The length of %s is %d\n",str,length);  
   
 // Accept two strings to compare  
 printf("\nEnter two strings for string compare :");  
 scanf("%s%s",str1,str2);  
   
   
 // Invoke string compare function to compare the str1 and str2 strings  
 comp\_res=str\_compare(str1,str2);  
   
 if (comp\_res < 0)  
 {  
 printf("String \"%s\" is less than string \"%s\"\n",str1,str2);  
 }  
 else if (comp\_res == 0)  
 {  
 printf("String \"%s\" is same as string \"%s\"\n",str1,str2);  
 }  
 else  
 {  
 printf("String \"%s\" is greater than string \"%s\"\n", str1,str2);  
 }  
   
 // Accept two strings for string concatenation  
 printf("\nEnter two strings for string concatenation :");  
 scanf("%s%s",str\_des,str\_src);  
   
 // Invoke string concatenation function   
 str\_concat(str\_des,str\_src);  
   
 // Print the concatenated string  
 printf("The string after concatenation is \"%s\"\n", str\_des);  
   
 return 1;  
 }  
   
 int str\_length(char s[])  
 {  
 int i;  
 for(i=0;s[i]!='\0';i++);  
 return i;  
 }  
   
   
 int str\_compare(char s1[], char s2[])  
 {  
 int i,j;  
 for(i=0,j=0;(s1[i] != '\0' && s2[j] != '\0');i++,j++)  
 {  
 if (s1[i] != s2[j])  
 {  
 return (s1[i] - s2[j]);  
 }  
 }  
 if (s1[i] == '\0' && s2[j] == '\0')  
 {  
 return 0;  
 }  
 else if(s1[i] == '\0' || s2[i] == '\0')  
 {  
 return (s1[i] - s2[i]);  
 }  
   
 }  
   
   
 void str\_concat(char s1[], char s2[])  
 {  
 int i,j;  
 for(i=0;s1[i] != '\0';i++);  
   
 for(j=0;s2[j] != '\0';i++,j++)  
 {  
 s1[i] = s2[j];  
 }  
 s1[i] = '\0';  
 }  
   
  
/\* Sample Output:  
1.  
$ cc prog10.c  
$ ./a.out  
  
Enter a string :rainbow  
The length of rainbow is 7  
  
Enter two strings for string compare :rain  
rainbow  
String "rain" is less than string "rainbow"  
  
Enter two strings for string concatenation :rain  
bow  
The string after concatenation is "rainbow"  
  
2.  
$ ./a.out  
  
Enter a string :thinkpad  
The length of thinkpad is 8  
  
Enter two strings for string compare :thinkpad  
think  
String "thinkpad" is greater than string "think"  
  
Enter two strings for string concatenation :think  
pad  
The string after concatenation is "thinkpad"  
\*/

**QUESTIONS FOR VIVA**

1. What is string concatenation?
2. How does string compare take place?
3. What is the terminating character while counting string length?
4. What is the length of a string?

## 

## Practical #11

**AIM**

Implementation of Bubble Sorting Technique

**PROBLEM STATEMENT**

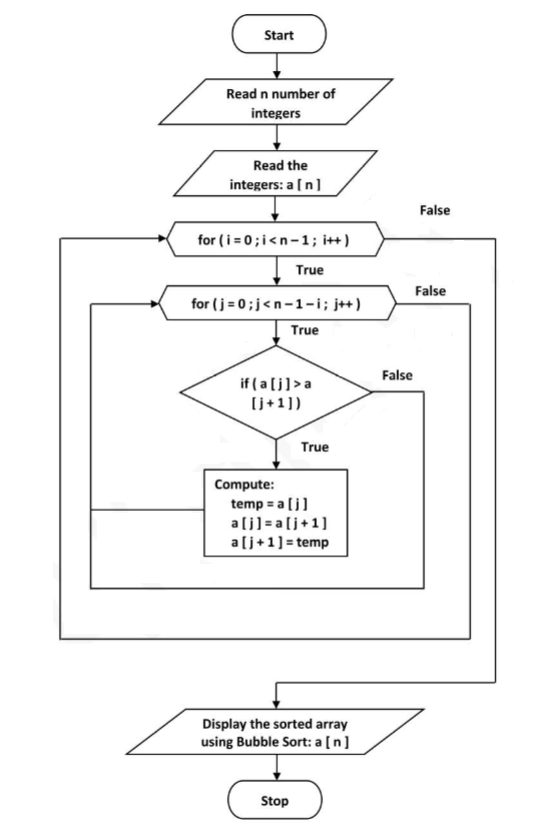
Develop an algorithm, implement and execute a C program that reads N integer numbers and arrange them in ascending order using BUBBLE SORT

**ALGORITHM**

Input: Size of the array and elements to be sorted

Output: List of elements in sorted order1. Start  
2. Read the size of array:a[n]  
3. Read the elements of array:n  
4. Sort the elements in array using bubble sort  
5. Display the elements of array in sorted order:a[n]  
6. Read the key to be searched:search\_key  
7. Initialize:  
 First=0  
 Last=n-1  
8. Check while first is less than or equal to last.If true goto step 9 else goto step 15  
9. Compute:  
 mid= (first+last)/2  
 res=cmp (a [mid], search\_key)  
10. Check if res is equal to zero.If true goto step 11 else goto step 12  
11. Display element found at position:mid and goto step 16  
12. Check if res is greater than zero.If true goto step 13 else goto step 14  
13. Compute:  
 Last=mid-1 goto step 8  
14. Compute:  
 First=mid+1 goto step 8  
15. Display key not found  
16. Stop

**FLOWCHART**

****

**PROGRAM**

/\*   
 \* Program 11:  
 \* Develop a program to sort the given set of N numbers using Bubble sort.  
 \*/  
   
 #include<stdio.h>  
   
 int main()  
 {  
 int a[20],n,i,j,temp;  
   
 // Accept array size  
 printf("Enter the number of elements :");  
 scanf("%d",&n);  
   
 // Accept n elements into the array  
 printf("Enter %d integers :",n);  
 for(i=0;i<n;i++)  
 {  
 scanf("%d",&a[i]);  
 }  
   
 // Sort the array elements  
   
 for(i=0;i<n-1;i++)  
 {  
 for(j=0;j<n-1-i;j++)  
 {  
 if(a[j] > a[j+1])  
 {  
 temp=a[j];  
 a[j]=a[j+1];  
 a[j+1]=temp;  
 }  
 }  
 }  
   
 // print the sorted array  
 printf("The sorted array is ....\n");  
 for(i=0;i<n;i++)  
 {  
 printf("%d\t",a[i]);  
 }  
   
 printf("\n");  
 return 0;  
 }  
   
**SAMPLE OUTPUT**  
1.  
$ cc prog11.c  
$ ./a.out  
Enter the number of elements :5  
Enter 5 integers :1  
3  
4  
2  
5  
The sorted array is ....  
1 2 3 4 5   
  
2.  
$ ./a.out  
Enter the number of elements :6  
Enter 6 integers :23  
55  
10  
-10  
0  
-400  
The sorted array is ....  
-400 -10 0 10 23 55   
  
  
**QUESTIONS FOR VIVA**

1. What is bubble sorting?
2. How does nested for loop works?
3. What are the passes in a bubble sorting ethod?

## Practical #12

**AIM**

Develop a program to find the square root of a given number N and execute for all possible inputs with appropriate messages. Note: Don’t use library function sqrt(n).

**PROBLEM STATEMENT**

Design and develop a flowchart to find the square root of a given number N. Implement a C program for the same and execute for all possible inputs with appropriate messages. Note: Don’t use library function: sqrt(n)

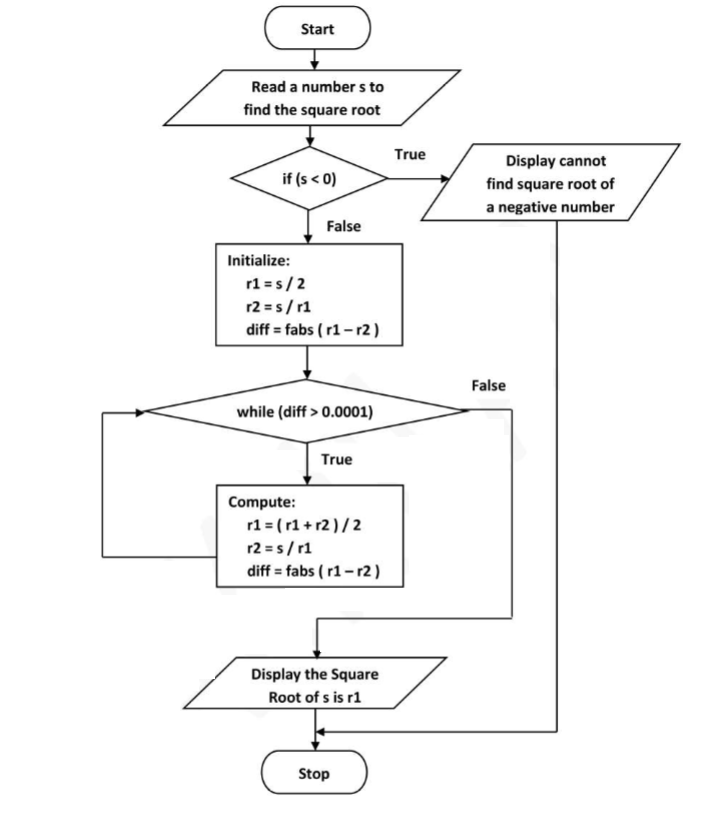
**ALGORITHM**

**Input:** A positive number

**Output:** To compute square root of the number

1. START
2. Read the number num
3. Check if the number is less than zero. If true goto Step 4 else goto Step 5
4. Display error and goto step 9
5. Initialize:
   1. r1 = num/2
   2. r2 = num/r1
   3. diff = fabs (r1 - r2)
6. Check if difference is greater than 0.0001. If true goto Step 7 else goto Step 8
7. Compute:
   1. r1 = (r1 + r2) / 2.0
   2. r2 = num / r1
   3. Diff = fabs (r1 - r2)
8. Display the sqrt of num is r1
9. STOP

**FLOWCHART**

****

**PROGRAM**

/\* Program 12:  
 \* Develop a program to find the square root of a given number N and execute for  
 \* all possible inputs with appropriate messages. Note: Don’t use library  
 \* function sqrt(n).  
 \*/  
   
#include <stdio.h>  
#include <math.h>  
   
int main ()  
{  
 float num,r1,r2,diff;   
   
 // Input the number to find the square root  
 printf("Please enter a positive integer: ");  
 scanf("%f", &num);  
   
 // Check if the number entered is negative, as we cannot find  
 // the square root of a negative number  
   
 if (num<0)  
 {  
 printf("\nCannot find the square root of a negative number\n\n");  
 return (1);  
 }  
   
 // Guess an initial value for root  
   
 r1 = num/2;  
 r2 = num/r1;  
 diff = fabs(r1-r2);  
   
 // Iterate to compute the root till error is not within tolerance  
 while (diff > 0.00001)  
 {  
 /\* Since on the number line guessed\_root and guessed\_root\_opp  
 always lie on the opposites of the actual root, we take the  
 average of the two as the new guessed root \*/  
 r1=(r1+r2)/2.0;  
   
 r2=num/r1;  
   
 diff=fabs(r1-r2);  
 }  
   
   
 // Dispaly the square root of the given number  
 printf("\nThe square root of %f is %f\n\n",num,r1);  
   
 return 0;  
}  
  
**Sample Output :**  
  
1.  
$cc prog12.c  
$ ./a.out  
Please enter a positive integer: 25  
  
The square root of 25.000000 is 5.000000  
  
2.  
$ ./a.out  
Please enter a positive integer: 2  
  
The square root of 2.000000 is 1.414216  
  
3.  
$ ./a.out  
Please enter a positive integer: 3  
  
The square root of 3.000000 is 1.732051  
  
4.  
$ ./a.out  
Please enter a positive integer: 10  
  
The square root of 10.000000 is 3.162278

**QUESTIONS FOR VIVA**

1. What is #define?
2. What is the difference between fabs(), fabsl(), fabsf(), abs()?
3. Why should we not use abs() with floating point variables?
4. How should we compare two floating point variables for equality?
5. What do you mean by error tolerance or round off error?
6. How does do-while loop work?
7. What is the difference between while and do while loops?

## PRACTICAL #13

**AIM**

Implement array of structures to read, write, compute average- marks and the students scoring above and below the average marks for a class of N students.

**PROBLEM STATEMENT**

Write a C program to maintain a record of n student details using an array of structures with fields to store student id, name, marks of subjects and the average. Assume appropriate data type for each field. Print the marks of the student scoring above and below average.

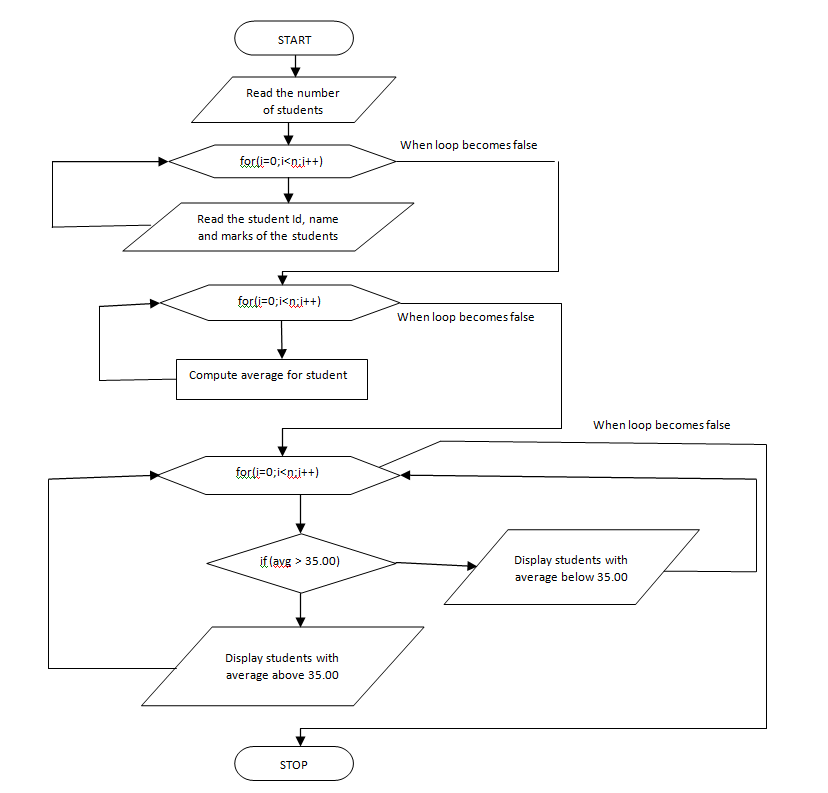
**ALGORITHM**

**Input:** Student details such as student id, name and marks

**Output:** To print the details of those students scoring above and below average

1. START
2. Read the number of students
3. For each student, read the student id, name and marks for all subjects.
4. Calculate the average of the marks and store in the avg field.
5. Print results.
6. Initialise loop
7. Read the average of each student
8. Check if avg>35.00
9. If yes print result else go to next iteration
10. Initialise loop
11. Read average of each student
12. Check if avg<35.00
13. If yes print result else go to next iteration
14. STOP

**FLOWCHART**

****

**PROGRAM**

/\* Program 13:  
 \* Implement structures to read, write, compute average marks and the   
 \* students scoring above and below the average marks for a class   
 \* of N students.  
 \*/  
   
 #include<stdio.h>  
   
 int main()  
 {  
 struct student  
 {  
 int stu\_id;  
 char name[20];  
 float lang1\_marks;  
 float lang2\_marks;  
 float sc\_marks;  
 float mat\_marks;  
 float sst\_marks;  
 float comp\_marks;  
 float avg;  
 };  
   
 struct student s[20];  
 int i,n;  
   
 // Accept the number of records/students  
 printf("Enter the number of records :");  
 scanf("%d",&n);  
   
 // Accept data for all the fields/members of each record  
 printf("Enter %d student details...\n",n);  
   
 for(i=0;i<n;i++)  
 {  
 printf("\n\nEnter student ID :"); // Student ID  
 scanf("%d",&s[i].stu\_id);  
   
 printf("Enter student name :"); // Studet Name  
 scanf("%s",s[i].name);  
   
 printf("Enter lang1 Marks:"); // Language 1 marks  
 scanf("%f",&s[i].lang1\_marks);  
   
 printf("Enter lang2 Marks :"); // Language 2 Marks  
 scanf("%f",&s[i].lang2\_marks);  
  
 printf("Enter science Marks :"); // Science Marks  
 scanf("%f",&s[i].sc\_marks);  
  
 printf("Enter Maths Marks :"); // Maths Marks  
 scanf("%f",&s[i].mat\_marks);  
  
 printf("Enter SST Marks :"); // SST Marks  
 scanf("%f",&s[i].sst\_marks);  
  
 printf("Enter Computer Marks :"); // Computer Marks  
 scanf("%f",&s[i].comp\_marks);  
 }  
   
 // Compute the average of each student  
 for(i=0;i<n;i++)  
 {  
   
 s[i].avg=(s[i].lang1\_marks + s[i].lang2\_marks + s[i].sc\_marks + s[i].mat\_marks + s[i].sst\_marks + s[i].comp\_marks)/6.0;  
 }  
   
 // Display student ID, name and average of all students   
 // who have scored above average marks  
 printf("Students scoring above the average marks....\n");  
 printf("\n\nID\tName\tAverage\n\n");  
   
 for(i=0;i<n;i++)  
 {  
 if(s[i].avg>=35.0)  
 printf("%d\t%s\t%f\n",s[i].stu\_id,s[i].name,s[i].avg);  
 }  
   
 // Display student ID, name and average of all students   
 // who have scored below average marks  
   
 printf("\n\nStudents scoring below the average marks....\n");  
 printf("\n\nID\tName\tAverage\n\n");  
   
 for(i=0;i<n;i++)  
 {  
 if(s[i].avg<35.0)  
 printf("%d\t%s\t%f\n",s[i].stu\_id,s[i].name,s[i].avg);  
 }  
   
 return 0;  
 }  
   
   
SAMPLE OUTPUT

Enter the number of records :5  
Enter 5 student details...  
  
  
Enter student ID :101  
Enter student name :manjula  
Enter lang1 Marks:56  
Enter lang2 Marks :54  
Enter science Marks :34  
Enter Maths Marks :32  
Enter SST Marks :45  
Enter Computer Marks :67  
  
  
Enter student ID :102  
Enter student name :kavitha  
Enter lang1 Marks:90  
Enter lang2 Marks :99  
Enter science Marks :98  
Enter Maths Marks :97  
Enter SST Marks :96  
Enter Computer Marks :100  
  
  
Enter student ID :103  
Enter student name :banu  
Enter lang1 Marks:54  
Enter lang2 Marks :22  
Enter science Marks :1  
Enter Maths Marks :12  
Enter SST Marks :12  
Enter Computer Marks :3  
  
  
Enter student ID :104  
Enter student name :pallavi  
Enter lang1 Marks:3  
Enter lang2 Marks :44  
Enter science Marks :5  
Enter Maths Marks :55  
Enter SST Marks :53  
Enter Computer Marks :21  
  
  
Enter student ID :105  
Enter student name :lalitha  
Enter lang1 Marks:54  
Enter lang2 Marks :2  
Enter science Marks :34  
Enter Maths Marks :56  
Enter SST Marks :78  
Enter Computer Marks :9  
Students scoring above the average marks....  
  
  
ID Name Average  
  
101 manjula 48.000000  
102 kavitha 96.666664  
105 lalitha 38.833332  
  
  
Students scoring below the average marks....  
  
  
ID Name Average  
  
103 banu 17.333334  
104 pallavi 30.166666

**QUESTIONS FOR VIVA**

1. What is the size of a structure?
2. What is the difference between array and structures?
3. What is the syntax for creating a structure?
4. What is a structure?
5. How do you declare a structure variable?
6. How to access a structure variable?
7. What is typedef?

## PRACTICAL #14

**AIM**

Develop a program using pointers to compute the sum, mean and standard deviation of all

elements stored in an array of n real numbers.

**PROBLEM STATEMENT**

Write a C program using pointers to compute the sum, mean and standard deviation of all elements stored in an array of n real numbers.

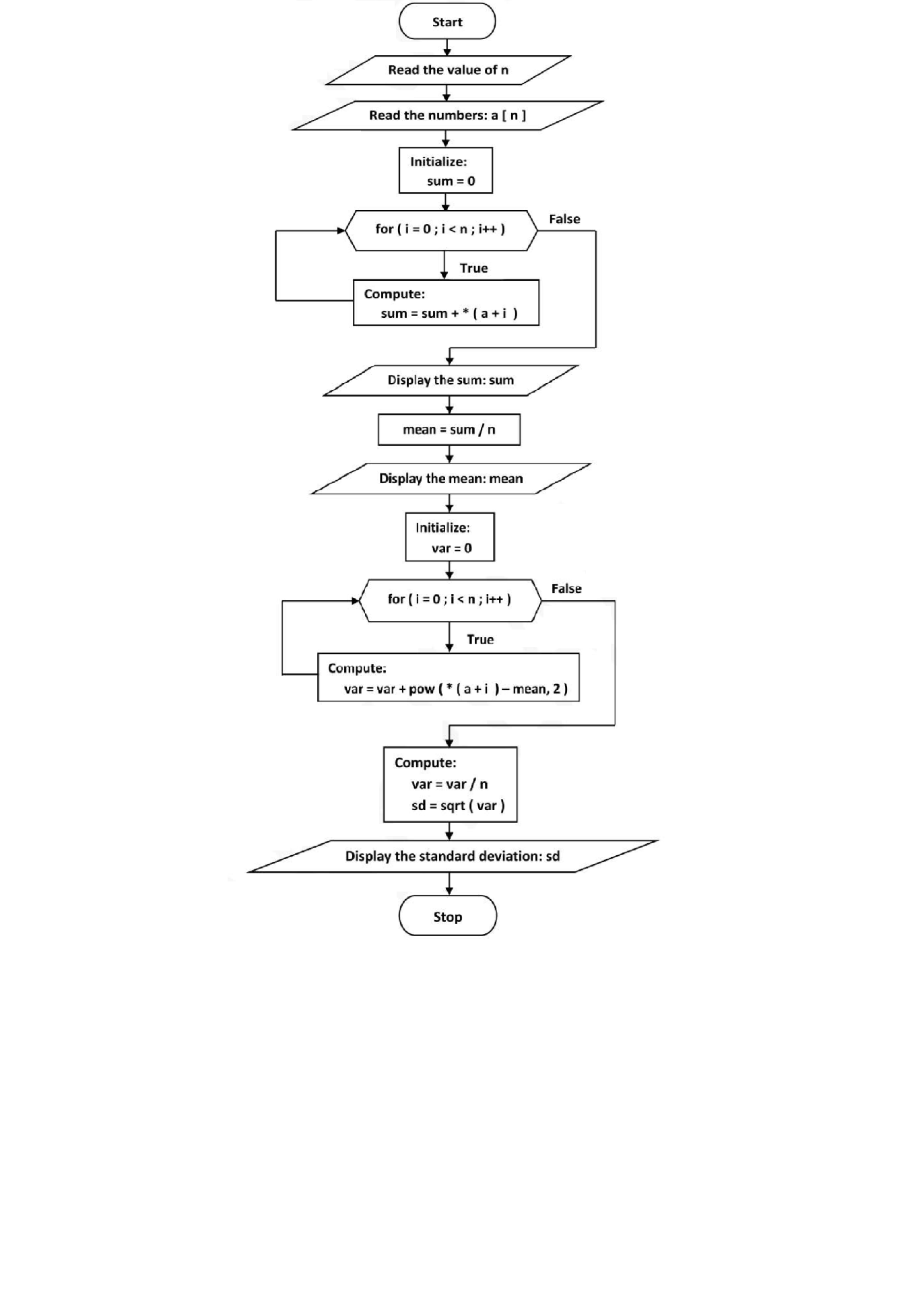
**ALGORITHM**

**Input:**An array of numbers

**Output:** To compute sum, mean and standard deviation

1. START
2. Read the value of n
3. Read the numbers into the array ‘a’ using the pointer ‘p’.
4. Initialise: i=0 and sum=0
5. Iterate the loop and perform the following
   1. Sum = sum + \*p
   2. Increment i
6. Display the sum
7. Calculate the mean using the formula mean = sum/n
8. Display mean
9. Initialise i=0 and var=0
10. Iterate the loop and perform the following
    1. var=var+pow((\*p-mean),2);
    2. Increment i
11. Display the variance
12. Compute
    1. var = var/n
    2. Sd = sqrt(var)
13. Display sd
14. STOP

**FLOWCHART**

****

**PROGRAM**

/\*   
 \* Program 14  
 \* Develop a program using pointers to compute the sum, mean and   
 \* standard deviation of all elements stored in an array of n real numbers.  
 \*/  
  
#include<stdio.h>  
#include<math.h>  
  
int main()  
{  
 int i,n;  
 float a[10],mean,sd,sum,var;  
 float \*p; // p is a pointer to float value  
   
 printf("\n Enter Number of elements :");  
 scanf("%d",&n);  
 printf("\n Enter the elements :");  
 p=a; // pointer p points to first element of a  
 for(i=0;i<n;i++)  
 {  
 scanf("%f",p);  
 p++; // pointer p points to the next element of the array  
 }  
   
 p=a; // Initialize p to the first element of the array  
 printf("\n input Elements are:\n");  
 for(i=0;i<n;i++)  
 {  
 printf("%f",\*p);  
 p++; // Pointer p is made to point to the next element  
 }  
   
 p=a; // Initialize p to the first element of the array  
   
 sum=sd=mean=var=0;  
   
 // Find the sum of the array elements  
 for(i=0;i<n;i++)  
 {  
 sum=sum+(\*p);  
 p++;  
 }  
 // Find the mean  
 mean=sum/n;  
   
 // Find variance  
 p=a;  
 for(i=0;i<n;i++)  
 {  
 var=var+pow((\*p-mean),2);  
 p++;  
 }  
 var=var/n;  
   
 // Find Standard Deviation  
 sd=sqrt(var);  
   
 // Print Sum, mean and Standard Deviation  
 printf("\n\n mean=%f\nsum=%f\nsd=%f\nvar=%f\n",mean,sum,sd,var);  
 return 0;  
}

**SAMPLE OUTPUT**

1.  
$ cc prog14.c -lm  
$ ./a.out  
  
 Enter Number of elements :5  
  
 Enter the elements :1  
2  
3  
4  
5  
  
The Sum=15.000000  
  
The mean=3.000000  
  
The Standard Deviation=1.414214  
  
2.  
$ ./a.out  
  
 Enter Number of elements :3  
  
 Enter the elements :55  
88  
22  
  
The Sum=165.000000  
  
The mean=55.000000  
  
The Standard Deviation=26.944387

**QUESTIONS FOR VIVA**

1. What is a pointer?
2. How to declare a pointer variable?
3. What is dangling pointer?
4. What is a pointer to pointer?
5. What is pointer to array?
6. What is pass by reference?

## Practical #15

**Aim**: Implement Recursive functions for Binary to Decimal Conversion. **#include<stdio.h>  
  
 int btod(int); // Prototype of btod() function  
   
 int main()  
 {  
 int binary,decimal;  
   
 // Accept input in binary format   
 printf("Enter binary input :");  
 scanf("%d",&binary);  
   
 // Invoke btod() to convert binary to decimal  
 decimal=btod(binary);  
   
 // Print the decimal equivalent of binary  
 printf("Decimal equivalent = %d\n", decimal);  
   
 return 0;  
 }  
  
// Function Definition of btod() function   
 int btod(int bin)  
{  
 if (bin==0)  
 {  
 return 0;  
 }   
 else  
 {  
 return (bin%10 + btod(bin/10) \* 2);  
 }  
}  
  
Sample Output:  
$ cc prog15.c  
$ ./a.out  
Enter binary input :01  
Decimal equivalent = 1  
  
$ ./a.out  
Enter binary input :100  
Decimal equivalent = 4  
  
$ ./a.out  
Enter binary input :1001  
Decimal equivalent = 9  
  
$ ./a.out  
Enter binary input :1101  
Decimal equivalent = 13  
  
$ ./a.out  
Enter binary input :110011001  
Decimal equivalent = 409  
\*/**

## Practical #16

**Aimt:** Program to read from name.txt and usn.txt and write the contents  
of both the files onto output.txt.

**#include<stdio.h>  
  
int main()  
{  
 FILE \*fp1,\*fp2,\*fp3;  
   
 char name[20],usn[20];  
   
 fp1=fopen("name.txt","r");  
 if(fp1==NULL)  
 {  
 printf("Error in opening file - name.txt in read mode...\n");  
 return 1;  
 }  
  
 fp2=fopen("usn.txt","r");  
 if(fp2==NULL)  
 {  
 printf("Error in opening file - usn.txt in read mode...\n");  
 return 2;  
 }  
   
 fp3=fopen("output.txt","w");  
 if(fp3==NULL)  
 {  
 printf("Error in opening file - output.txt in write mode...\n");  
 return 3;  
 }  
   
 fprintf(fp3,"Name\tUSN\n");  
   
 // read from name.txt and usn.txt  
 while (fscanf(fp1,"%s",name)!= EOF && fscanf(fp2,"%s",usn)!= EOF)  
 {  
 fprintf(fp3,"%s\t%s\n",name,usn); // write to output.txt  
 }  
   
 // Close all the file pointers  
 fclose(fp1);  
 fclose(fp2);  
 fclose(fp3);  
   
 // open output.txt in read mode  
 fp3=fopen("output.txt","r");  
 if(fp3==NULL)  
 {  
 printf("Error in opening file - output.txt in read mode...\n");  
 return 4;  
 }  
   
 // read from output.txt  
 while(fscanf(fp3,"%s\t%s\n",name,usn) != EOF)  
 {  
 printf("%s\t%s\n",name,usn); // write on to output screen  
 }  
 fclose(fp3);  
 return 0; // return success  
}**