Jadavpur University

Computer Science and Numerical Methods Practical Record File Session 2022-26

NAME : ***SHAHIR HABIB***

YEAR : UG - 1

DEPT : CSE

ROLL : 002210501006

SUB : Computer Science and Numerical Methods

Table Of Contents

ASSINGMENT-1

Flowchart and Algorithm

1. **Write an algorithm to determine the maximum of three numbers. Also draw the corresponding flow chart.**

ALGORITHM :

1. Ask the user to enter three integer values.
2. Read the three integer values in num1, num2, and num3 (integer variables).
3. Check if num1 is greater than num2.
4. If true, then check if num1 is greater than num3.
    a. If true, then print 'num1' as the greatest number.
    b. If false, then print 'num3' as the greatest number.
5. If false, then check if num2 is greater than num3.
    a. If true, then print 'num2' as the greatest number.
    b. If false, then print 'num3' as the greatest number.

2. **Write an algorithm to determine the sum of individuat digits of a given integer. Also draw the corresponding flowchart.**

ALGORITHM :

1: Get number by user

2: Get the modulus/remainder of the number

3: sum the remainder of the number

4: Divide the number by 10

3. Write an algorithm to print the reverse of a number read as input. Also draw the corresponding flow chart.

ALGORITHM :

1. Ask the user to enter any number.

2. Declare and initialize another variable reversed with 0, where reversed an integer variable.

3. Get the last digit of the given number by performing the modulo division (%) and store the value in last_digit variable, likey last_digit= number % 10.

4. Multiply reversed by 10 and add last_digit, like reversed = reversed*10 + last_digit.

5. Divide numbered by 10, like numbered/10.

6. Repeat the steps 3 to 5 till numbered is not equal to (or greater than) zero.5: Repeat the step 2 while number is greater than 0.

4. Write an algorithm to determine whether a given number is prime or not. Also draw the corresponding flowchart.

ALGORITHM:

Step 1 - Take integer variable A

Step 2 → Divide the variable A with (A-1 to 2)

Step 3 → If A is divisible by any value (A-1 to 2) it is not prime

Step 4 → Else it is prime

5. Write an algorithm to generate the first 100 prime numbers. Also draw the corresponding flow chart.

ALGORITHM:

1.Assingn the counter c to be 0

2. Starting from number 2 increase 1 to check if the are prime

3. Lets say if a integer is A

4. Divide the integer by A-1 to2

5. If A is divisible by any value in between it is not prime

6. If it is divisible by any value in between it is prime so increase the value of counter by 1

7. Again increase 1 and check for A+1 by repeating Step 3 and 4

8. Repeat the whole step from 1 to 7 till the value of counter become equal to 100.

6. Draw a flowchart to input three numbers in the variables a, b and c and hence to find the roots of the quadratic equation ax' + bx + c= O. Consider carefully the zero input values of the coefficients a, b and c.

ALGORITHM:

1.Ask the user to enter the values of a,b and c.

2.Repeat step1 if a=0.

3.Assing the value of b*b -4a*c to D

3.Root of quadratic equation will be (-b + sqrtD)/2*a

and other will be (-b+ sqrtD)/2*a.

ASSINGMENT-2

CONSOLE I/O AND CONDITIONAL STATEMENTS

1. Write a C program that reads two values from the keyboard, swaps their values and prints out the result.

C CODE:

```
#include <stdio.h>

void swap(int a,int b){
    int t;
    t=a;
    a=b;
    b=t;
    printf("After swappimg a: %d and b : %d ",a,b);
}
int main(){
    int a,b;
    printf("Enter the first number a :\n");
    scanf("%d",&a);
    printf("Enter the second number b :\n");
    scanf("%d",&b);
    swap(a,b);
}
```

OUPUT:

```
Enter the first number a :

09

Enter the second number b :

32

After swappimg a: 32 and b : 9
```

2. **If a three-digit integer is input through the keyboard, write a program to calculate the sum of its digits.**
(Hint: Use the modulo operator 'W) .

C CODE :

```
#include <stdio.h>

int main()
```

```
        {
        int s=0,n;
         printf("Enter the value of n :\n");
         scanf("%d",&n);
         while(n>0){
         s=s+(n%10);

         n=n/10;

         }
         printf("the sum of digit is %d",s);
         return 0;
        }
```

OUTPUT :

```
Enter the value of n :
546
the sum of digit is 15
```

3. Input two integer numbers and divide the larger number by the smaller one. Then display the result using printf() function as a fractional number first and then as a real valued number.

(Example: 9 divided by 5 shall yield 4/5" and "1.8" respectively,)

C CODE:

```
#include<stdio.h>

int main(){
    int a,b;
    float f;
    printf("Enter the number a :\n");
    scanf("%d",&a);
    printf("Enter the number b :\n");
    scanf("%d",&b);
    if(b)
    f=(float)a/b;
    printf("The divided numbers in integer form :%d/%d\n",a,b);
    printf("The divided numbers in decimal form :%0.3f\n",f);
}
```

OUTPUT:

```
Enter the number a :
9
Enter the number b :
5
The divided numbers in integer form :9/5
The divided numbers in decimal form :1.800
```

4. Write a C program which accepts basic salary as input and prints the gross salary, which is the sum of the basic, dearness allowance (60% of basic salary), and house rent allowance (15% of basic salary).

C CODE:

```
#include <stdio.h>
int main()
{
    int b, g, d, r;
    printf("Enter the basic salary  :");
    scanf("%d", &b);
    d = 0.6 * b;
    r = .15 * b;
    g = d + b + r;
    printf("Your gross salary which is sum of basic
salary : %d \n ,dear allowance 60%% of basic
salary:%d\n and rent allowance 15%% of basic salary:
%d\n is \n %d", b, d, r, g);
    return 0;
}
```

OUTPUT:

```
Enter the basic salary1000
Your gross salary which is sum of basic salary : 1000
,dear allowance 60% of basic salary:599
and rent allowance 15% of basic salary: 149  is
 1748
```

5. Any year is input through the keyboard, Write a program to determine whether the year is a leap year or not, (Hint: Use the % (modulus) operator)

C CODE:

```
#include <stdio.h>
int main()
```

```
    {
        int y;
        printf("Enter the year :");
        scanf("%d", &y);

        if (y % 4 == 0)
        {
            if (y % 100 == 0)
            {
                if (y % 400 == 0)
                    printf("leap year");
                else
                    printf("🛇 not leap year");
            }
            else
                printf("leap year");
        }
        else
            printf("not 🛇 leap year");
        }
```

```
OUTPUT:
Enter the year :4000
leap year
Enter the year : 2022
Not a leap year
```

6. Write a program to check whether a triangle is valid or not, when (i) the three angles of the triangle are entered through the Keyboard (ii) three sides of the triangle are entered through the keyboard.

C CODE:

```
    #include <stdio.h>

    int main()
    {
        int a, b, c, sum, A, B, C;
        printf("Enter the angles and side :");
        scanf("%d %d %d %d %d %d", &a, &b, &c, &A, &B, &C);
        sum = a + b + c;
        if (sum == 180 && ((A < B + C) && (B < A + C) && (C <
    A + B)))
            printf("valid 📐");
        else
            printf("not valid triangle");

        return 0;
    }
OUTPUT:
    Enter the angles and side :7 8 3 90 60 30
    not valid triangle
```

7. Given three points (xl, yl), (x2, y2) and (x3, y3), write a program to check if the three points fall on one straight line

C CODE :

```
#include <math.h>
#include <stdio.h>
 int main()
{
    int x1, x2, x3, y1, y2, y3;
    float m;
    printf("Enter the point x1,y1,x2,y2,x3,y3\n");
    scanf("%d %d %d %d %d %d", &x1, &y1, &x2, &y2, &x3, &y3);
    m = (float)(y1 - y2) / (x1 - x2);
    if (m == (float)(y3 - y1) / (x3 - x1))
        printf("points on line");
    else
        printf("points not on line");
    return 0;
}
OUTPUT:
    Enter the point
    0 0 1 1 2 2
    points on line
```

8. Given the coordinates (x, y) of a centre of a circle and its radius, write a program which will determine whether a point lies inside the circle, on the circle or outside the circle.

(Hint: #incjude Use sqrt( ) and pow( ) functions)

C CODE

```
#include <stdio.h>
#include <math.h>
int main()

{
    int x1, x2, y1, y2, r, m;
    m = sqrt((x1 - x2) * (x1 - x2)) + sqrt((y1 - y2) * (y1 - y2));
    printf("enter the centre coordinate ");
    scanf("%d %d", &x1, &y1);
    printf("enter the radius");
    scanf("%d,&r");
     printf("enter the point to check if it is on the circle
     out of circle or in the circle \n");
    scanf("%d %d", &x2, &y2);
    if (r > m)
    {
        printf("point is inside the circle ");
    }
    else if (r == m)
    {
        printf("point is on the circle ");
    }
    else
        printf("points outside the circle ");
    return 0;
}
```

OUTPUT:

```
enter the centre coordinate 0 0
enter the radius5
enter the point to check if it is on the circle out of
circle or in the circle
3 4
point is inside the circle
```

9. **Any character is entered through the keyboard, write a program to determine whether the character entered is a capital letter, a small case letter, a digit or a special symbol**.

C CODE:

```
#include <stdio.h>
#include <ctype.h>
int main()
{
    char a;
    int b;
    printf("enter the character you want to check\n");
    scanf("%c", &a);
    b = toascii(a);
    if ((b >= 'a') && (b <= 'z'))
    {
        printf("small case letter");
    }
    else if ((b >= 'A') && (b <= 'Z'))
    {
        printf("Capital letter");
    }
    else if ((b >= '0') && (b <= '9'))
        printf("integer number");

    else
        printf("special symbol");
    return 0;
}
```

```
OUTPUT:
enter the character you want to check
*
special symbol
enter the character you want to check
a
small case letter
enter the character you want to check
Y
```

```
    Capital letter
```

10. Given as input an integer number of seconds, write a program to print as output the equivalent time in hours, minutes and seconds. Recommended output format is something like 7322 seconds is equivalent to 2 hours 2 minutes 2 seconds.

C CODE:

```
    #include <stdio.h>
    int main()
    {
        int s, h, m;
        printf("enter the time in seconds : ");
        scanf("%d", &s);
        m = s / 60;
        h = m / 60;
        s = s % 60;
        m = m % 60;
        printf("Time : %d hours %d minutes %d seconds", h, m, s);
    }
```

OUTPUT:

```
    enter the time in seconds : 5386
    Time : 1 hours 29 minutes 46 seconds
```

ASSINGMENT-3

LOOPS

1. Write a C program which accepts a number n and prints

a. all integers divisible by n between 1 and 100 where value of n is provided by the user.

C CODE :

```
#include <stdio.h>
int main()
{
    int n;
    printf("Enter the number n :\n");
    scanf("%d", &n);
    printf("Numbers divisible by %d are \n", n);
    for (int i = n; i < 100; i++)
    {
        if ((i % n) == 0)
        {
            printf("%d  ", i);
        }
    }
    return 0;
}
```

OUTPUT:

```
Enter the number n :
7
Numbers divisible by 7 are
7 14 21 28 35 42 49 56 63 70 77 84 91 98
```

b. all prime numbers between 1 and n.

C CODE

```
#include <stdio.h>
int ifprime(int n)
{
```

```
                if (n == 2)
                {
                    return 1;
                }
                int i;
                for (i = 2; i < n - 1; i++)
                {
                    if ((n % i) == 0)
                    {
                        return 0;
                    }
                }
                return 1;
            }
            int main()
            {
                int n;
                printf("Enter the number n :\n");
                scanf("%d", &n);
                printf("Prime number between 1 and %d are\n", n);
                for (int i = 2; i <= n; i++)
                {
                    if (ifprime(i))
                    {
                        printf("%d  ", i);
                    }
                }
                return 0;
            }
```

```
OUTPUT:
Enter the number n :
99
Prime number between 1 and 99 are
2  3  5  7  11  13  17  19  23  29  31  37  41  43  47  53  59
61  67  71  73  79  83  89  97
```

c. all prime factors of n.

C CODE

```
#include <stdio.h>
int ifprime(int n)
{
    if (n == 2)
    {
        return 1;
    }
    int i;
    for (i = 2; i < n - 1; i++)
    {
        if ((n % i) == 0)
        {
            return 0;
        }
    }
    return 1;
}
int main()
{
    int n,k;
    printf("Enter the number n :\n");
    scanf("%d", &n);
    for (int i = 2; i < n; i++)
    {
        k = n;
        if ((n % i) == 0)
        {
            if (ifprime(i))
            {
                while ((k%i)== 0)
                {
                    printf("%d  ", i);
```

```
                                    k=k/i;
                                }
                            }
                        }
                    }
                    return 0;
                }
```

OUTPUT :
Enter the number n :

```
64

2 2 2 2 2 2
```

d. octal equivalent of n

C CODE

```
#include <stdio.h>
int decimal_toOcta(int n){
    int rem;
    int result=0;
    int power=0;
    while(n>0){
        rem=n%8;
        n=n/8;
        result= result +rem*pow(10,power);
        power++;
    }
    return result;
}
int main(){
    int n;
printf("Enter the number : ");
scanf("%d",&n);
decimal_toOcta(n);
return 0;
}
OUTPUT: Enter the number : 164
243
```

e. sum of digits.

C CODE

```
#include <stdio.h>
int sod(int n){
    int s=0;
    while(n>0){
        s=s+(n%10);
        n=n/10;
    }
    return s;
}
int main(){
   int n,k;
    printf("Enter the number n :\n");
    scanf("%d",&n);
    printf("Sum of digits of number is :%d",sod(n));
return 0;
}
```

OUTPUT:

```
Enter the number n :  5638

Sum of digits of number is : 22
```

f. factorial of n.

C CODE

```
#include <stdio.h>
int fact(int n){
    if(n<2){
        return n;
    }
    return (n*fact(n-1));
}
int main(){
int n,k;
    printf("Enter the number n :\n");
    scanf("%d", &n);
```

```
            printf("(%d)!=%d",n,fact(n));
        return 0;
        }
```

OUTPUT:

```
    Enter the number n : 5
    (5)!=120
```

g. reverse of n.

C CODE

```
        #include <stdio.h>
        int rev(int n){
        3    int s=0;
            while(n>0){
            s= s*10 +(n%10);
            n=n/10;
            }
            return s;
        }
        int main(){
        int n;
            printf("Enter the number n : ");
            scanf("%d", &n);
            printf("Reverse of given number is : %d",rev(n));
        return 0;
        }
```

OUTPUT:

```
        Enter the number n : 4252
        Reverse of given number is : 2524
```

2. Write a C program to find out the sum of the following series.

a. S=1+2+3+4+ … +n

C CODE:

```
    #include <stdio.h>
    int main(){
    int n,s=0;
        printf("Enter the number n : ");
        scanf("%d", &n);
        for(int i=1;i<=n;i++){
            s=s+i;
        }
        printf("sum of digits 1+2+3+...   + %d is %d",n,s);
    return 0;
    }
```

OUTPUT:

```
    Enter the number n : 9
    sum of digits 1+2+3+...   + 9 is 45
```

b. S=1.2+2.3+3.4+4.5+ … +n.(n+1)

C CODE

```
    #include <stdio.h>
    int main(){
    int n,s=0;
        printf("Enter the number n : ");
        scanf("%d", &n);
        for(int i=1;i<=n;i++){
            s=s+i*(i+1);
        }
        printf("sum of digits 1.2+2.3+3.4+...   + %d.%d+1 is
    %d",n,n+1,s);
    return 0;
    }
    OUTPUT:
    Enter the number n : 5
    sum of digits 1.2+2.3+3.4+...   + 5.6+1 is 70
```

c. S=1!+2!+3!+4!+ … +n!

C CODE

```
    #include <stdio.h>
    int fact(int n){
        if(n<2){
            return n;
        }
        return (n*fact(n-1));
    }
    int main(){
    int n,s=0;
        printf("Enter the number n : ");
        scanf("%d", &n);
        for(int i=1;i<=n;i++){
            s=s+fact(i);
        }
        printf("sum of digits 1!+2!+3!+...   + %d! is
    %d",n,s);
    return 0;
    }
```

OUTPUT:

```
     Enter the number n : 5
     sum of digits 1!+2!+3!+...   + 5! is 153
```

d. S = 1@ + 2@ + 3@ + 4@ + ... + n@

where, n@ is the sum of all factors of n. Example: 6@ = 1+2+3+6 = 12

C CODE

```
    #include <stdio.h>
    int main(){
    int n,s=0;
        printf("Enter the number n : ");
        scanf("%d", &n);
        for(int j=1;j<=n;j++){
            for(int i=1;i<=n;i++){
            if((j%i)==0){
                s=s+i;
            }
            }
        }
        printf("Sum of 1@ +2@ +3@+...  + %d@ is %d ",n,s);
    return 0;
```

```
}
OUTPUT:
Enter the number n : 4
Sum of 1@ +2@ +3@+...  + 4@ is 15
```

3. **Write a program to generate all combinations of digit 1, 2 and 3 using a for loop.**

C CODE

```
#include <stdio.h>
int main()
{
    int i, j, k;
    printf("All combinations of digit 1,2,3 are \n");
    for (i = 1; i <= 3; i++)
    {
        for (j = 1; j <= 3; j++)
        {
            if (i != j)
            {
                for (k = 1; k <= 3; k++)
                {
                    if (k != i && k != j)
                    {
                        printf("%d %d %d    ", i, j, k);
                    }
                }
            }
        }
    }
}
```

5. **Write a program named SINE to find the sine of an angle. The angle and its unit (degree, radian or grade) should be provided as command line arguments. For the units, short forms as d/D (for degree), r/R (for radian) or g/G (for grade) may be used. The program should use the series $sin(x) = x - x^3/_{3!} + x^5/_{5!} \cdots$ for evaluation. Take care of negative angles and angles in all the quadrants.**

C CODE:

```c
#include <stdio.h>
#include <math.h>
int fact(int n)
{
    if ((n == 0) || (n == 1))
        return 1;
    else
        return n * fact(n - 1);
}
int main()
{
    float r, d, g, s = 0.0;
    int i;
    char a;
    printf("Enter the angle you want to find sine of \n Enter
G for gradian system \n D for degree and \n R for radian\n");
    scanf("%c", &a);
    switch (a)
    {
    case 'G':
        printf("gradian system :");
        scanf("%f", &g);
        r = (g * 1.11) * (180 / 3.1415);
        break;
    case 'D':
        printf("Degree system :");
        scanf("%f", &d);
        r = (180 / 3.14159) * d;
        break;
    case 'R':
        printf("Radian system :");
        scanf("%f", &r);
        break;
    }
    for (i = 1; i < 10; i = i + 2)
    {
        switch (i)
        {
```

```
        case '1':
            s = s + pow(r, i) / fact(i);
            break;
        case '5':
            s = s + pow(r, i) / fact(i);
            break;
        case '9':
            s = s + pow(r, i) / fact(i);
        default:
            s = s + pow(r, i) / fact(i);
            break;
        }
    }
    printf("the sine of given angle is = %f", s);
    return 0;
}
```

6. Write a C program to print the first n numbers of the Fibonacci sequence. The Fibonacci sequence is constructed by adding the last two numbers of the sequence so far to get the next number in the sequence. The first and second numbers of the sequence are defined as 0 and 1. We get:

0, 1, 1, 2, 3, 5, 8, 13, 21…

CODE:

```
    #include <stdio.h>
    int fibonaci(int n)
    {
        if (n < 2)
        {
            return n;
        }
        else
            return (fibonaci(n - 1) + fibonaci(n - 2));
    }
    int main()
    {
        int n;
        printf("enter the limit of fibonacci series : ");
        scanf("%d",&n);
        printf("fibonacci series upto %d terms is :",n);
        for(int i=0;i<=n;i++){
            printf("%d  ",fibonaci(i));
        }
        return 0;
    }
```

OUTPUT:

```
enter the limit of fibonacci series : 7
fibonacci series upto 7 terms is :0  1  1  2  3  5  8  13
```

7. **Write a program to print out all Armstrong numbers between 1 and 500. If the sum of cubes of each digit of the number is equal to the number itself, then the number is called an Armstrong number. For example, 153= $1^3 + 5^3 + 3^3$.**

C CODE:

```
        #include <stdio.h>
        #include<math.h>
        int isArms(int n){
            int s=0,k;
            k=n;
            while(n){
                s=s+ pow((n%10),3);
                n=n/10;
            }
            if(s==k){
                return 1;
            }
            else
            return 0;
        }
        int main(){
        printf("Armstrong number between 1 and 500 are : ");
        for(int i=2;i<500;i++){
            if(isArms(i)){
                printf("%d  ",i);
            }
        }
        return 0;
        }
```

OUTPUT:

```
    Armstrong number between 1 and 500 are : 153   370   371   407
```

8. Write a C program which prints the first 10 happy numbers. If you iterate the process (assume maximum 100 iterations) of summing the squares of the decimal digits of a number and if the process terminates in 1, then the original number is called a Happy number. For example 7 is a happy number as 7 → 49 → 97 → 130 → 10 → 1.

C CODE:

```
            #include <stdio.h>
            int main()
            {
                int n, i, r, temp, sum;
            printf("Happy Numbers are\n");
            for (i = 1; i < 100; i++)
            {
                n = i;
                sum = 0;
                temp = n;
                while (sum != 1 && sum != 4)
                {
                    sum = 0;
                    while (n > 0)
                    {
                        r = n % 10;
                        sum = sum + (r * r);
                        n = n / 10;
                    }
                    n = sum;
                }
                n = temp;
                if (sum == 1)
                    printf("%d ", n);
            }
            return 0;
             }
```

OUTPUT: Happy Numbers are

```
1 7 10 13 19 23 28 31 32 44 49 68 70 79 82 86 91 94 97
```

9. An important property of square numbers: If a natural number is a square number, then it has to be the sum of Successive Odd Numbers starting from 1.

For ex

Perfect Squares	Sum of odd numbers
4	1+3
9	1+3+5
16	1+3+5+7
25	1+3+5+7+9
36	1+3+5+7+9+11
49	1+3+5+7+9+11+13

```c
#include <stdio.h>
int main(){
int n,s=0,c=0;
printf("Enter any perfect square : ");
scanf("%d",&n);
for(int i=1;i<n;i=i+2){
    s=s+i;
    c++;
    if(s==n){
        printf("Square root of %d is %d",n,c);
        break;
    }
}
return 0;
}
```

```
OUTPUT:
Enter any perfect squar : 256
Square root of 256 is 16
```

10. Write the c program that print the following pattern for the input n=4. The value of n input by the user.

PATTERN 1

```
1
1 2
```

```
1 2 3

1 2 3 4
```

C CODE :

```
#include <stdio.h>
int main(){
int i,j,n;
n=4;
for(int i=1;i<=n;i++){
    for(int j=1;j<=i;j++){
    printf("%d ",j);
    }
printf("\n");
}
return 0;
}
```

PATTERN 2

```
   1

  1 2

 1 2 3

1 2 3 4
```

C CODE:

```
#include <stdio.h>
int main(){
int i,j,n;
n=4;
for(int i=1;i<=n;i++){
for(int j=n-i;j>=0;j--){
    printf(" ");
}
for(int k=1;k<=i;k++){
    printf("%d",k);
}
printf("\n");
}
return 0;
}
```

PATTERN 3

```
   1
  121
 12321
1234321
```

C CODE:

```
#include <stdio.h>

int main(){
int i,j,k,n=4;
for(i=1;i<=n;i++){
    for(j=1;j<=n-i;j++){
        printf(" ");
    }
    for(k=1;k<=i;k++){
        printf("%d",k);
    }
    for(int l=k-2;l>0;l--){
        printf("%d",l);
    }
    printf("\n");
}
return 0;
}
```

PATTERN 4

```
   1
   11
  121
  1221
 12321
 123321
1234321
12344321
```

C CODE :

```
#include<stdio.h>
int main(){
    int n=4;
    for(int i=1;i<=n;i++){
        for(int j=n-i;j>=0;j--){
            printf(" ");
        }

        int inc=1;
        for(int j=1;j>0;j+=inc){
            printf("%d",j);
            if(j==i)
            inc=-1;
        }
        printf("\n");
        for(int j=n-i;j>=0;j--){
            printf(" ");
        }

        inc=1;
        for(int j=1;j>0;j+=inc){
            printf("%d",j);
            if(j==i)
            {inc=-1;
            printf("%d",j);}
        }
        printf("\n");
    }
     }
```

PATTERN 5

43210

321

2

C CODE:

```
#include<stdio.h>
int main(){
    int st=4;
    int end=0;
    while(st>=end){
        for(int i=st;i>=end;i--){
            printf("%d",i);
        }
        st--;
        end++;
        printf("\n");
    }
}
```

PATTERN 6

```
   +
  +++
 +++++
+++++++
 +++++
  +++
   +
```

C CODE

```
    #include <stdio.h>
    int main(){
        int n=4;
        for(int i=1;i<=n;i++){
            for(int k=1;k<=n-i;k++){
            printf(" ");
            }
        for(int j=1;j<=(2*i-1);j++){
            printf("+");
        }
        printf("\n");
        }
        n--;
        for(int i=1;i<=n;i++){
            for(int j=1;j<=i;j++){
                printf(" ");
            }
            for(int k= 1;k<=2*(n-i)+1; k++){
                printf("+");
            }
            printf("\n");
        }
    return 0;
    }
```

PATTERN 7

```
   1
  121
 12321
1234321
 12321
  121
   1
```

C CODE

```
#include <stdio.h>
int main(){
int i,j,k,n=4;
for(i=1;i<=n;i++){
    for(j=1;j<=n-i;j++){
        printf(" ");
    }
    for(k=1;k<=i;k++){
        printf("%d",k);
    }
    for(int l=k-2;l>0;l--){
        printf("%d",l);
    }
    printf("\n");
}
n--;
```

```
        for(int i=1;i<=n;i++){
            for(int j=1;j<=i;j++){
                printf(" ");
            }
            for(int k=1;k<=n-i+1;k++){
                printf("%d",k);
            }
            for(int l=n-i;l>0 ;l--){
                printf("%d",l);
            }
            printf("\n");
        }
        return 0;
```

PATTERN 8

```
        +++++++++
        ++++ ++++
        +++   +++
        ++     ++
        +       +
        ++     ++
        +++   +++
        ++++ ++++
        +++++++++
```

C CODE

```
        #include <stdio.h>
        int main(){
        int n=4;
        for(int i=1;i<=2*n+1;i++){
            printf("+");
        }
        printf("\n");
        for(int i=1;i<=n;i++){
            for(int j=1;j<=n-(i-1);j++){
                printf("+");
            }
            for(int k=1;k<=2*(i-1)+1;k++){
                printf(" ");
            }
             for(int j=1;j<=n-(i-1);j++){
                printf("+");
            }
            printf("\n");
        }
        n--;
        for(int i=1;i<=n;i++){
            for(int j=1;j<=i+1;j++){
                printf("+");
            }
            for(int k=1 ;k<2*(n-i+1) ;k++){
                printf(" ");
            }
            for(int j=1;j<=i+1;j++){
                printf("+");
            }
            printf("\n");
        }
        for(int i=1;i<=2*(n+1)+1;i++){
            printf("+");
        }
        return 0;
        }
```

PATTERN 9

```
    +
   + +
  +   +
 +     +
  +   +
   + +
    +
```

C CODE

```
#include <stdio.h>
int main(){
    int n=4;
for(int i=1;i<=n;i++){
    for(int j=n-i;j>0;j--){
        printf(" ");
    }
    printf("+");
    for(int k=2*i -3;k>0;k--){
        printf(" ");
    }
    if(i!=1){
    printf("+");
    }
    printf("\n");
}
n--;
for(int i=1;i<=n;i++){
    for(int j=1;j<=i;j++){
        printf(" ");
    }
    printf("+");
    for(int k=1;k<=n-2*(i-1);k++){
        printf(" ");
    }
    if(i!=3){
    printf("+\n");
    }
}
return 0;
}
```

ASSINGMENT-4

ARRAY

1. Write a program in C to reverse the contents Of the elements Of an integer array.

C CODE :

```
#include <stdio.h>
void reverseArray(int a[],int n){
    int t;
    for(int i=0;i<n/2;i++){
        t=a[i];
        a[i]=a[n-1-i];
        a[n-1-i]=t;
    }
}
void print_1DArray(int a[],int n){
    for(int i=0;i<n;i++){
        printf("a[%d]=%d    ",i+1,a[i]);
    }
}
void scan_1DArray(int a[],int *p){
if((*p)==0){
printf("Enter the length of array : ");
scanf("%d",p);
}
printf("Enter the contents of array\n");
for(int i=0;i<(*p);i++){
    printf("a[%d] = ",i+1);
    scanf("%d",&a[i]);
}
}
int main(){
    int a[15],n=0;
    scan_1DArray(a,&n);
    reverseArray(a,n);
    print_1DArray(a,n);
}
```

OUTPUT:

```
Enter the length of array : 7

Enter the contents of array
```

```
a[1] = 5

a[2] = 87

a[3] = 21

a[4] = 389

a[5] = 32

a[6] = 89

a[7] = 25


a[1]=25         a[2]=89          a[3]=32          a[4]=389
a[5]=21            a[6]=87           a[7]=5
```

2.Write a program in C to read n number Of values in an array. After that, count the total number Of duplicate elements in that array. Then copy the elements except the duplicate elements Of that array into another array and display this array in reverse order.

C CODE:

```
#include <stdio.h>

int main()
{
    int k, n, d = 0, a[10], b[10] = {}, i, j, x = 0, p, c
= 0;
    printf("enter the value of n \n");
    scanf(" %d", &n);
    printf("enter the integers \n");
    for (i = 0; i < n; i++)
    {
        scanf("%d", &a[i]);
    }
    for (int i = 0; i < n; i++)
    {
        k = a[i];
        for (p = i + 1; p < n; p++)
        {
            if (k == a[p])
            {
                c++;
                d++;
                break;
```

```
                    }
                }
                if (d == 0)
                {
                    b[x] = k;
                    x++;
                }
                d = 0;
            }
            printf("Number of duplicate element %d \n", c);
            for (i = 9; i >= 0; i--)
            {
                if (b[i] != 0)
                {
                    printf("%d ", b[i]);
                }
            }
            return 0;
             }
```

```
        OUTPUT: enter the value of n
        7
        enter the integers
        32
        54
        98
        32
        54
        67
        12
        Number of duplicate element 2
12 67 54 32 98
```

3.Write a menu-driven program for accepting values in two square matrices Of 3x3 dimension and generate their sum, difference and product.

C CODE:

```
        #include <stdio.h>
        void scan2DArray(int arr[3][3],int a,int b){
        printf("Enter the elements row wise : ");
        for(int i=0;i<a;i++){
```

```
        for(int j=0;j<b;j++){
            printf("arr[%d][%d]= ",i+1,j+1);
            scanf("%d", & arr[i][j]);
        }
        printf("\n");
    }
    }
    void print2DArray(int arr[3][3],int a,int b){
        for(int i=0;i<a;i++){
            for(int j=0;j<b;j++){
                printf("a[%d][%d]= %d\t",i+1,j+1,arr[i][j]);
            }
            printf("\n");
        }
    }
    int main(){
    int arr[3][3],p[3][3];
    int brr[3][3],s[3][3];
    int d[3][3];
    int a=3,b=3;
    scan2DArray(arr,3,3);
    scan2DArray(brr,3,3);
     for(int i=0;i<a;i++){
            for(int j=0;j<b;j++){
                s[i][j]=arr[i][j]+brr[i][j];
            }
     }
     for(int i=0;i<a;i++){
            for(int j=0;j<b;j++){
                d[i][j]=arr[i][j]-brr[i][j];
            }
     }
     for(int i=0;i<a;i++){
            for(int j=0;j<b;j++){
                p[i][j]=0;/*important step*/
                for(int k=0;k<3;k++){
                    p[i][j]+=arr[i][k]*brr[k][j];
                }
            }
     }
     printf("First Matrix a is : \n");
    print2DArray(arr,3,3);
    printf("Second Matrix is \n ");
```

```
          print2DArray(brr,3,3);
          printf("The Sum  of Matrices is\n");
          print2DArray(s,3,3);
          printf("The Product of Matrices is\n");
          print2DArray(p,3,3);
          printf("The Difference of Matrices is\n");
          print2DArray(d,3,3);
          return 0;
          }
```

OUTPUT:

```
Enter the elements row wise : arr[1][1]= 1

arr[1][2]= 2

arr[1][3]= 3

arr[2][1]= 4

arr[2][2]= 5

arr[2][3]= 6

arr[3][1]= 7

arr[3][2]= 8

arr[3][3]= 9


Enter the elements row wise : arr[1][1]= 1

arr[1][2]= 0

arr[1][3]= 0

arr[2][1]= 0

arr[2][2]= 1

arr[2][3]= 0

arr[3][1]= 0

arr[3][2]= 0

arr[3][3]= 1


First Matrix a is :

a[1][1]= 1      a[1][2]= 2      a[1][3]= 3

a[2][1]= 4      a[2][2]= 5      a[2][3]= 6
```

```
a[3][1]= 7      a[3][2]= 8      a[3][3]= 9
Second Matrix is
 a[1][1]= 1     a[1][2]= 0      a[1][3]= 0
a[2][1]= 0      a[2][2]= 1      a[2][3]= 0
a[3][1]= 0      a[3][2]= 0      a[3][3]= 1
The Sum  of Matrices is
a[1][1]= 2      a[1][2]= 2      a[1][3]= 3
a[2][1]= 4      a[2][2]= 6      a[2][3]= 6
a[3][1]= 7      a[3][2]= 8      a[3][3]= 10
The Product of Matrices is
a[1][1]= 1      a[1][2]= 2      a[1][3]= 3
a[2][1]= 4      a[2][2]= 5      a[2][3]= 6
a[3][1]= 7      a[3][2]= 8      a[3][3]= 9
The Difference of Matrices is
a[1][1]= 0      a[1][2]= 2      a[1][3]= 3
a[2][1]= 4      a[2][2]= 4      a[2][3]= 6
a[3][1]= 7      a[3][2]= 8      a[3][3]= 8
```

4. Write a program to find the range Of a set of integers entered by the user. Range is the difference between the smallest and biggest number in the list.

CODE:

```
#include <stdio.h>
void scan_1DArray(int a[],int *p){
if((*p)==0){
printf("Enter the number of integer: ");
scanf("%d",p);
}
printf("Enter the numbers of array\n");
for(int i=0;i<(*p);i++){
    printf("a[%d] = ",i+1);
    scanf("%d",&a[i]);
}
}
int main(){
int arr[15],n=0;
scan_1DArray(arr,&n);
int min =arr[0];
int max =arr[0];
for(int j=0;j<n;j++){
    if(arr[j]>max){
        max=arr[j];
    }
    if(arr[j]<min){
        min=arr[j];
    }
}
printf("Range of entered values is : %d",max-min);
return 0;
}
```

OUTPUT:

```
        Enter the numbers of integr : 5
        Enter the numbers of array
        a[1] = 34
        a[2] = 89
        a[3] = 07
        a[4] = 55
        a[5] = 98
        Range of entered values is : 91
```

5. Write a C program which accepts ten integers from the user and prints them in ascending order use an array to store the integers.

C CODE:

```
        #include <stdio.h>
        void scan_1DArray(int a[],int *p){
        if((*p)==0){
        printf("Enter the number of integers : ");
        scanf("%d",p);
        }
        printf("Enter the integers \n");
        for(int i=0;i<(*p);i++){
            printf("a[%d] = ",i+1);
            scanf("%d",&a[i]);
        }
        }
        void print_1DArray(int a[],int n){
            for(int i=0;i<n;i++){
                printf("a[%d]=%d    ",i+1,a[i]);
            }
        }
        void bubble_sort(int x[],int n)
        {
            int i,j,t;
            for(i=n-1;i>0;i--){
                for(j=0;j<i;j++){
                    if( x[j]>x[j+1]){
                        t=x[j];
                        x[j]=x[j+1];
                        x[j+1]=t;
                    }
                }
            }
        }
        int main(){
        int n=0,a[10];
        scan_1DArray(a,&n);
        bubble_sort(a,n);
        print_1DArray(a,n);
        return 0;
        }
```

```
OUTPUT:
Enter the number of integers : 5
Enter the integers
a[1] = 65
a[2] = 87
a[3] = 321
a[4] = 65
a[5] = 54
a[1]=54    a[2]=65    a[3]=65    a[4]=87    a[5]=321
```

6.Write a C program which accepts roll numbers Of ten students and marks obtained by them in five subjects and prints the names Of the students who have obtained highest and second highest marks subject wise.

C CODE:

```
#include <stdio.h>
int main(){
    char nam[10][10];
    int marks[10][5],k,p,max[10]={},sma=0;
    char sub[5][10]={

{"CPNM"},{"ELCTRICAL"},{"ELCTRONIC"},{"ED"},{"MATH"}
    };
    printf("Enter the names of student along with their marks \n");
for(int i=0;i<10;i++){
    printf("student %d: ",i+1);
    scanf("%s",nam[i]);
    for(int j=0;j<5;j++){
        printf("Enter the marks for %s : ",sub[j]);
        scanf("%d",&marks[i][j]);
    }
}
for(int i=0; i<5 ; i++){
    for(int j=0; j<10 ; j++){
        if(max[i]<marks[j][i]){
            max[i]=marks[j][i];
            k=j;
        }
}
printf("Highest marks in %s is of %s \n",sub[i],nam[k]);
}
for(int i=0; i< 5; i++){
    for(int j=0; j< 10; j++){
        if((sma<marks[j][i])&&(max[i]>marks[j][i])){
            sma=marks[j][i];
            p=j;
        }
}
printf("Second Hingest mars in %s is of %s \n",sub[i],nam[p]);
sma=0;
}
return 0;
}
          OUTPUT:
          student 1: swapnamoy
          Enter the marks for CPNM : 24
          Enter the marks for ELCTRICAL : 15
          Enter the marks for ELCTRONIC : 20
          Enter the marks for ED : 29
          Enter the marks for MATH : 47
          student 2: abhi
          Enter the marks for CPNM : 21
          Enter the marks for ELCTRICAL : 24
          Enter the marks for ELCTRONIC : 30
          Enter the marks for ED : 29
          Enter the marks for MATH : 25
          student 3: abrar
          Enter the marks for CPNM : 21
          Enter the marks for ELCTRICAL : 30
          Enter the marks for ELCTRONIC : 27
          Enter the marks for ED : 28
          Enter the marks for MATH : 26
          student 4: shahir
          Enter the marks for CPNM : 20
          Enter the marks for ELCTRICAL : 36
          Enter the marks for ELCTRONIC : 24
          Enter the marks for ED : 28
          Enter the marks for MATH : 20
          student 5: masud
          Enter the marks for CPNM : 26
          Enter the marks for ELCTRICAL : 24
          Enter the marks for ELCTRONIC : 25
          Enter the marks for ED : 34
          Enter the marks for MATH : 29
          student 6: saifool
          Enter the marks for CPNM : 20
          Enter the marks for ELCTRICAL : 30
          Enter the marks for ELCTRONIC : 15
          Enter the marks for ED : 45
          Enter the marks for MATH : 16
          student 7: jagannath
          Enter the marks for CPNM : 32
          Enter the marks for ELCTRICAL : 24
          Enter the marks for ELCTRONIC : 24
          Enter the marks for ED : 25
          Enter the marks for MATH : 29
          student 8: akshat
          Enter the marks for CPNM : 26
          Enter the marks for ELCTRICAL : 27
          Enter the marks for ELCTRONIC : 38
          Enter the marks for ED : 20
          Enter the marks for MATH : 23
          student 9: krishti
          Enter the marks for CPNM : 27
          Enter the marks for ELCTRICAL : 26
          Enter the marks for ELCTRONIC : 23
          Enter the marks for ED : 12
          Enter the marks for MATH : 49
          student 10: jeetesh
          Enter the marks for CPNM : 30
          Enter the marks for ELCTRICAL : 29
          Enter the marks for ELCTRONIC : 23
          Enter the marks for ED : 20
          Enter the marks for MATH : 27
          Highest marks in CPNM is of jagannath
          Highest marks in ELCTRICAL is of abrar
          Highest marks in ELCTRONIC is of akshat
          Highest marks in ED is of saifool
          Highest marks in MATH is of krishti
          Second Hingest mars in CPNM is of jeetesh
          Second Hingest mars in ELCTRICAL is of shahir
          Second Hingest mars in ELCTRONIC is of abhi
          Second Hingest mars in ED is of masud
          Second Hingest mars in MATH is of swapnamoy
```

7.Write a C program which accepts a matrix and prints its transpose.

C CODE:

```
 #include <stdio.h>

void scan2DArray(int arr[10][10],int a,int b){
printf("Enter the elements row wise : ");
for(int i=0;i<a;i++){
    for(int j=0;j<b;j++){
        printf("a[%d][%d]= ",i+1,j+1);
        scanf("%d", & arr[i][j]);
    }
    printf("\n");
}
}
void print2DArray(int arr[10][10],int a,int b){
    for(int i=0;i<a;i++){
        for(int j=0;j<b;j++){
            printf("a[%d][%d]=
%d\t",i+1,j+1,arr[i][j]);
        }
        printf("\n");
    }
}
int main(){
    int m,n,a[10][10],b[10][10];
printf("Enter dimension of matrix(in mxn form) : ");
scanf("%dx%d",&m,&n);
scan2DArray(a,m,n);
for(int i=0;i<m;i++)
for(int j=0;j<n;j++){
    b[j][i]=a[i][j];
}
printf("matrix :\n");
print2DArray(a,m,n);
printf("transpose of matrix :\n");
print2DArray(b,n,m);
return 0;
```

```
        }
    OUTPUT:
        Enter the elements row wise : a[1][1]= 1
        a[1][2]= 2
        a[1][3]= 2

        a [2][1]= 4
        a[2][2]= 5
        a [2][3]= 6

        matrix :
        a[1][1]= 1      a[1][2]= 2      a[1][3]= 2
        a[2][1]= 4      a[2][2]= 5      a[2][3]= 6
        transpose of matrix :
        a[1][1]= 1      a[1][2]= 4
        a[2][1]= 2      a[2][2]= 5
        a[3][1]= 2      a[3][2]= 6
```

8.Write a C program to replace a square matrix by its transpose without using a second matrix.

C CODE:

```
    #include <stdio.h>
    void scan2DArray(int arr[10][10],int a,int b){
    printf("Enter the elements row wise : ");
    for(int i=0;i<a;i++){
        for(int j=0;j<b;j++){
            printf("a[%d][%d]= ",i+1,j+1);
            scanf("%d", & arr[i][j]);
        }
        printf("\n");
    }
    }
    void print2DArray(int arr[10][10],int a,int b){
        for(int i=0;i<a;i++){
            for(int j=0;j<b;j++){
                printf("a[%d][%d]= %d\t",i+1,j+1,arr[i][j]);
            }
            printf("\n");
        }
    }
    int main(){
        int n,t,a[10][10];
    printf("Enter size of square matrix : ");
    scanf("%d",&n);
    scan2DArray(a,n,n);
    printf("matrix :\n");
    print2DArray(a,n,n);
    for(int i=0;i<n;i++)
    for(int j=0;j<n;j++){
        if(i<j){
            t=a[i][j];
            a[i][j]=a[j][i];
            a[j][i]=t;
        }
    }
    printf("transpose of matrix :\n");
    print2DArray(a,n,n);
    return 0;
    }
```

OUTPUT:

```
Enter size of square matrix : 3
Enter the elements row wise : ar[1][1]= 1
a[1][2]= 2
a[1][3]= 3

a[2][1]= 4
a[2][2]= 5
a[2][3]= 6

a[3][1]= 7
a[3][2]= 8
a[3][3]= 9
matrix :
a[1][1]= 1      a[1][2]= 2      a[1][3]= 3
a[2][1]= 4      a[2][2]= 5      a[2][3]= 6
```

```
a[3][1]= 7      a[3][2]= 8      a[3][3]= 9
transpose of matrix :
a[1][1]= 1      a[1][2]= 4      a[1][3]= 7
a[2][1]= 2      a[2][2]= 5      a[2][3]= 8
a[3][1]= 3      a[3][2]= 6      a[3][3]= 9
```

9. Consider the following procedure:

i. Take as input any four-digit number, using at least two different digits. (Leading zeros are

allowed.)

ii. Arrange the digits in descending and then in ascending order to get two four-digit numbers, adding leading zeros if necessary.

iii. Subtract the smaller number from the bigger number. Let the difference be the new four digit number.

iv. Go back to step ii.

The above process known as Kaprekar's routine, will always reach a fixed point (Known as Kaprekar Constant). Write a C Code to implement the algorithm given above and find out the constant number Also create an output file 'output data' in the working folder and write the following with appropriate format each step of iteration: The number, the larger number. the smaller number and the difference of the larger and the smaller number

Note: A. The fixed point is achieved when in two consecutive Steps the same number is obtained

B. In C the binary arithmetic operation m%n gives the remainder when m is divided by n.

C CODE:

```c
#include <stdio.h>
void bubble_sort(int x[],int n);
int kaprekar(int n,int p){
    p=n;
    int d[4],min=0,max=0,dif=0;
    for(int j=0;j<4;j++){
        d[j]=n%10;
        n=n/10;
    }
    bubble_sort(d,4);
```

```
        for(int i=0;i<4;i++){
        max= max*10+ d[3-i];
        min = min*10 +d[i];
    }
    dif=(max-min);
    if(dif==p){
        return dif;
    }
    else
    return kaprekar(dif,p);
    }
    void bubble_sort(int x[],int n)
    {
        int i,j,t;
        for(i=n-1;i>0;i--){
            for(j=0;j<i;j++){
                if( x[j]>x[j+1]){
                    t=x[j];
                    x[j]=x[j+1];
                    x[j+1]=t;
                }
            }
        }
    }

    int main(){
       int b,c;
       printf("Enter the 4 digit number with atleast 2
    different digits :\n");
       scanf("%d",&b);
    printf("The Kaprekar number is :%d",kaprekar(b,0));
    return 0;
    }
```

OUTPUT:

```
Enter the 4 digit number with atleast 2 different
digits:
1234
The Kaprekar number is :6174
```

10. Write a program which takes some numbers and computes the standard deviation of them. Formulas:

For a set Of n values. x_1,x_2,x_3,x_4 the average or mean is given by $\bar{x} = \frac{x_1+x_2}{n}\overline{+\cdots x_n}$

The standard deviation is given by: s= $\sqrt{\frac{\sum(xi-\bar{x})2}{n}}$

C CODE:

```
#include <stdio.h>
#include <math.h>
void scan_1DArray(int a[],int *p){
if((*p)==0){
printf("Enter the number of data : ");
scanf("%d",p);
}
printf("Enter the data \n");
for(int i=0;i<(*p);i++){
    printf("a[%d] = ",i+1);
    scanf("%d",&a[i]);
}
}
void print_1DArray(int a[],int n){
    for(int i=0;i<n;i++){
        printf("a[%d]=%d    ",i+1,a[i]);
    }
}
int main(){
int n=0,a[10];
scan_1DArray(a,&n);
print_1DArray(a,n);
float avg=0;
for(int i=0;i<n;i++){
    avg+=a[i];
}
avg=avg/n;
float s=0;
for(int j=0;j<n;j++){
```

```
            s+= pow((a[j]-avg),2);
        }
        s= sqrt(s/n);
        printf("the standard deviation of given numbers are : 
        %f",s);
        return 0;
    }
    OUTPOT:
    Enter the number of data : 5
    Enter the data
    a[1] = 42
    a[2] = 16
    a[3] = 7
    a[4] = 32
    a[5] = 9
    a[1]=42    a[2]=16    a[3]=7    a[4]=32    a[5]=9    the 
standard deviation of given numbers are : 13.614697
```

ASSINGMENT-5

FUNCTION AND POINTER

1. Write recursive functions for following tasks.

a. Binary equivalent of a number.

C CODE :

```
#include <stdio.h>

int decimal_toBinary(int n){

    int rem;
    int result=0;
    int power=0;
    while(n>0){
        rem=n%2;
        n=n/2;
        result= result +rem*pow(10,power);
        power++;
    }
    return result;
}
int main(){
    int n;
printf("Enter the decimal number :");
scanf("%d",&n);
return 0;
}
```

```
OUTPUT:    Enter the deciamal number : 8
           1000
```

b. Sum of individual digits of a number passed as argument.

C CODE :

```
        #include <stdio.h>
        int sod(int n){
            int s=0;
            while(n>0){
                s=s+(n%10);
                n=n/10;
            }
            return s;
        }
        int main(){
           int n,k;
            printf("Enter the number n :\n");
            scanf("%d",&n);
            printf("Sum of digits of number is :%d",sod(n));
        return 0;
        }
```

OUTPUT:

```
Enter the number n :

5638

Sum of digits of number is : 22
```

2. Write a C program using functions which accepts a string from the user and performs the following tasks.

a. Counts the number of characters in the string without using string library functions.

C CODE :

```
#include <stdio.h>
int count(char s[]){
    int i=0;
while(s[i]!='\0'){
    i++;
}
return i;
}
int main(){
    char s[80];
printf("Enter the string :\n");
scanf("%[^\n]s",s);
printf("Total number of characters in string are :
%d",count(s));
return 0;
}
OUTPUT:
Enter the string :
hello i am shahir
Total number of characters in string are : 17
```

b. Prints the reverse of the string without using string library functions.

C CODE :

```
#include <stdio.h>
void reverse_string(char s[],int n){
    for(int i=n-1;i>=0;i--){
        printf("%c",s[i]);
    }
}
int count(char s[]){
    int i=0;
while(s[i]!='\0'){
    i++;
}
return i;
}
int main(){
char s[80];
printf("Enter the string :\n");
scanf("%[^\n]s",s);
int n= count(s);
printf("The reverse of given string is: \n");
reverse_string(s,n);
return 0;
}
```

```
OUTPUT:
Enter the string :
hello i am shahir
The reverse of given string is:
rihahs ma i olleh
```

3.Write a C program which accepts a full name from the user prints the initials. Eg. SRT for Sachin Ramesh Tendulkar.

C CODE :

```
 #include <stdio.h>
int count(char s[]){
    int i=0;
while(s[i]!='\0'){
    i++;
}
return i;
}
int main(){
    char s[80];
printf("Enter the name :");
scanf("%[^\n]s",s);
int n=count(s);
printf("The initials of given name is : ");
printf("%c",s[0]);
for(int i=0;i<n;i++){
    if(s[i]==' '){
        printf("%c",s[i+1]);
    }
}
return 0;
}
```

OUTPUT :

```
Enter the name : Mahendra Singh Dhoni

The initials of given name is : MSD
```

4. Write a program to count the number of occurrences of any two vowels in succession in a line of text.

C CODE :

```
#include <stdio.h>

#include<ctype.h>
int count(char s[]){
    int i=0;
```

```
        while(s[i]!='\0'){
            i++;
        }
        return i;
        }
        int isvow(char c){
            char s= toupper(c);
            if(s=='A'|| s=='E'|| s=='I'||s=='O'|| s=='U')
            return 1;
            else
            return 0;
        }
        int vowcount(char s[],int n){
            int c=0;
            for(int i=0;i<n;i++){
                if(isvow(s[i]) && isvow(s[i+1])){
                    c++;
                }
            }
            return c;
        }
        int main(){
        char s[80];
        printf("Enter the string :\n");
        scanf("%[^\n]s",s);
        int n=count(s);
        printf("The number of successive vowels in the string are
        : %d",vowcount(s,n));
        return 0;
        }
```

OUTPUT:

```
Enter the string :

aeiou cjbdcEO fjkIoU

The number of successive vowels in the string are : 7
```

5. Write a program that converts (Do not use any string library function):

a. A string like “123” to integer 123.

C CODE :

```
#include<stdio.h>
int str_to_int(char num[]){
    int sum=0;
    int i=0;
    while(num[i]!='\0'){
        sum=sum*10 + (num[i]-48);
        i++;
    }
    return sum;
}
int main(){
printf("%d",str_to_int("123"));
}
```

```
OUTPUT: 123
```

b. An integer like 123 to string "123".

C CODE :

```
#include<stdio.h>

void num_to_str(int num){
    char str[100];
    int i=0;
    while(num>0){
        int rem=num%10;
        num=num/10;
        str[i]=rem+48;
        i++;
    }

    for(int j=i-1;j>=0;j--){
        printf("%c",str[j]);
    }
}
int main(){
num_to_str(123);
}
```

```
OUTPUT: 123
```

6. Write a C program which accepts a string from the user and performs the following tasks. (Do

not use any string library function.)

a. Check whether it is palindrome or not. [Example of a palindrome string: "abcba", abba"]

C CODE :

```
#include <stdio.h>
int count(char s[]){
    int i=0;
while(s[i]!='\0'){
    i++;
}
return i;
}
int main(){
    int r=1;
 char   s[80];
printf("enter the string\n");
scanf("%[^\n]s",s);
int n=count(s);
for(int c=0;c<n/2;c++){
    if(s[c] != s[n-c-1]){
        r=0;
        break;
    }
}
if(r)
printf("Entered text is palindrome ");
else
printf("Entered text is not palindrome");
return 0;
}
```

OUTPUT:

```
enter the string
able was i ere i saw elba
Entered text is palindrome
enter the string
false
Entered text is not palindrome
```

b. Counts the number of characters and words in it.

C CODE :

```
#include <stdio.h>
#include<ctype.h>
void scanline(char s[], int *v, int *co, int *d, int *sp, int *ch)
{
    char c;
    int i = 0;
    while ((c = toupper(s[i]))!= '\0')
    {
        if (c == 'A' || c == 'E' || c == 'I' || c == 'O' || c ==
'U')
            ++ *v;
        else if (c >= 'A' && c <= 'Z')
            ++ *co;
        else if (c >= '0' && c <= '9')
            ++ *d;
        else if (c >= ' ' || c <= '\t')
            ++ *sp;
        else
            ++*ch;
        i++;
    }
    return;
}
int main()
{
    char s[80];
    int vow = 0, conso = 0, dig = 0, space = 0, charecter = 0;
    printf("Enter the string :\n");
    scanf("%[^\n]s", s);
    scanline(s, &vow, &conso, &dig, &space, &charecter);
    printf("The number of vowel in the line : %d\n", vow);
    printf("The number of consonant in the line : %d\n", conso);
    printf("The number of space in the line : %d\n", space);
    printf("The number of charachter in the line : %d", charecter);
    return 0;
}
  OUTPUT:

   personal computer with memories in excess o 4096 kb are now
  quite common.

  The number of vowel in the line : 23

  The number of consonant in the line : 33

  The number of space in the line : 13

  The number of charachter in the line : 0
```

7. Write a program in C to store n numbers in an array and print the elements using pointers. Also compute the sum of all elements of that array using pointers.

C CODE :

```c
#include <stdio.h>
void scan_1DArray(int a[],int *p){
if((*p)==0){
printf("Enter the length of array : ");
scanf("%d",p);
}
printf("Enter the contents of array\n");
for(int i=0;i<(*p);i++){
    printf("a[%d] = ",i+1);
    scanf("%d",&a[i]);
}
}
void print_1DArray(int a[],int n){
    int s=0;
    for(int i=0;i<n;i++){
        printf("a[%d]=%d    ",i+1,*(a+i));
        s+=*(a+i);
    }
    printf("The sum of all elements is : %d",s);
}
int main(){
int a[10],n=0;
scan_1DArray(a,&n);
print_1DArray(a,n);
```

```
        return 0;
        }
OUTPUT: Enter the length of array : 5

Enter the contents of array

a[1] = 1

a[2] = 2

a[3] = 3

a[4] = 4

a[5] = 5

a[1]=1    a[2]=2    a[3]=3    a[4]=4    a[5]=5    The sum of all
elements is : 15
```

8. Write a C function which accepts a string str1 and returns a new string str2 which is str1 with each word reversed. Do not use any string library function.

C CODE :

```c
#include<stdio.h>
#include<string.h>
int char_count(char str[]){
    int i=0;
    while(str[i]!='\0'){
        i++;
    }
    return i;
    }
void rev(char str[],char res[],int st,int end){
    int j=end;
    for(int i=st;i<=end ;i++){
        res[j]=str[i];
        //printf("%c",res[j]);
        j--;
    }
    if(j!=-1)
    res[j]=' ';
}
void reverse_words_in_string(char str[]){
```

```
char s[150]="";
int st=0;
int end=0;
while(str[end]!='\0'){
    if(str[end]==' '){
        rev(str,s,st,end-1);
        st=end+1;
    }
    end++;
    if(str[end]=='\0'){
        rev(str,s,st,end-1);
        st=end+1;
    }
}
printf("%s",s);
}
int main(){
    char str[150];
    scanf("%[^\n]s",str);

reverse_words_in_string(str);
printf("\n");
}
```

```
  OUTPUT: hello i am shahir

  olleh i ma rihahs
```

9. Write a function squeeze(s,c) which removes all occurrences of the character c from the string s.

C CODE :

```
#include <stdio.h>
int count(char s[]){
    int i=0;
while(s[i]!='\0'){
    i++;
}
return i;
```

```
        }
        void squeez(char s[],char c){
        char b[80];
        int n=count(s),j=0;
        for(int i=0;i<n;i++){
            if(s[i]!='c'){
                b[j]=s[i];
                j++;
            }
        }
        printf("%s",b);
        }
        int main(){
            char s[80];
        printf("Enter the string :\n");
        scanf("%[^\n]s",s);
        squeez(s,'c');
        return 0;
    }
```

OUTPUT:

```
Enter the string :

caca co coffecio cano

aa o offeio ano
```

10. Write the function strend(s,t), which returns 1 if the string t occurs at the end of the string s, and zero otherwise.

C CODE :

Numerical Method

Practical - 1

Objective: Write a menu-driven program for finding roots of a nonlinear equation using Bisection, Regula Falsi and Newton-Raphson method.

Code:
```
#include<stdio.h>
#include<math.h>
float func(float x){
    return pow(x,2)-1;
}
float derivative(float (*func)(float),float x){
    float y1=(*func)(x);
    float y2=(*func)(x+0.0001);
    return (y2-y1)/0.0001;
}
void bisection(float g1,float g2,float e){
    float y1=func(g1);
    float y2 =func(g2);
    if (y1*y2>0){
        printf("root does not lie b/w the gusses");
        return;
    }
    else if(y1==0){
        printf("%f is a root",g1);
        return;
    }
    else if(y2==0){
        printf("%f is a root",g2);
        return;
    }
    int i=1;

    while(fabs(g2-g1)>e){
        float x=(g1+g2)/2;
        float y=func(x);
        printf("%-4d |%f   %f    %f    %f
\n",i,g1,g2,x,y);
        i++;
        if(y*y2<0){
            g1=x;
            g2=g2;
        }
        else if(y1*y<0){
            g1=g1;
            g2=x;
        }
        else if(y==0)
        {printf("%f is a root",x);
        return;}
    }
}

void falsepoint(float g1,float g2,float e){
    float y1=func(g1);
    float y2 =func(g2);
    if (y1*y2>0){
        printf("root does not lie b/w the gusses");
        return;
    }
    else if(y1==0){
        printf("%f is a root",g1);
        return;
    }
    else if(y2==0){
        printf("%f is a root",g2);
        return;
    }
    int i=1;
    while(fabs(y1)>e && fabs(y2)>e && i<100){
        float x=(g1+g2)/2;
        float y=func(x);
        printf("%-4d |%f   %f    %f    %f
\n",i,g1,g2,x,y);
        i++;
        if(y*y2<0){
            g1=x;
            g2=g2;
        }
        else if(y1*y<0){
            g1=g1;
            g2=x;
        }
        else if(y==0)
        {printf("%f is a root",x);
        return;}
    }
}

void Newton_rampson(float g1,float e){
    float u=derivative(func,g1);
    float prev_x=g1+1;
    int i=1;
    while(fabs(prev_x-g1)>e){
        prev_x=g1;
        float y1=func(g1);
        printf("%-4d |%f    %f \n",i,g1,y1);
        u=derivative(func,g1);
        g1=g1-(1/u)*(y1);
        i++;
    }
}
int main(){
char c;
scanf("%c",&c);
if(c=='B'||'b'){
    float g1;
    float g2;
    float e;
    scanf("%f %f %f",&g1,&g2,&e);
    bisection(g1,g2,e);
}
else if(c=='F'||'f'){
    float g1;
    float g2;
    float e;
    scanf("%f %f %f",&g1,&g2,&e);
    falsepoint(g1,g2,e);
}
else if(c=='B'){
    float g1;

    float e;
    scanf("%f %f",&g1,&e);
    Newton_rampson(g1,e);
}}
```

Output:

```
These are following choices:
        B/b for bisection method.
        F/f for bisection method.
        N/n for bisection method.
B
Enter g1, g2, e: 0.5 1.5 0.0001
1    |0.500000   1.500000    1.000000    0.000000
1.000000 is a root
```

Numerical Methods

Practical - 1

Objective: Write a menu-driven program for finding roots of a nonlinear equation using Bisection, Regula Falsi and Newton-Raphson method.

Code:

```
#include<stdio.h>
#include<math.h>
#define mx 10
#define epsilon 0.00001
int c[mx];int n;
float f(float x)
{
    return x*tan(x)-4;
}
void set()
{
    printf("enter the degree :");
    scanf("%d",&n);
   // int c[mx]; // n+1 coefficients
    printf("enter %d coefficients : \n ",n+1);
    for(int i=0;i<=n;i++)
    {
        scanf("%d",&c[i]);
    }
}
void bisec_method(float (*p)(float),float a,float b)
{
    float y1=(*p)(a);
    float y2=(*p)(b);
    float c=a,error=0.0;int i=0;
    if(y1*y2 >= 0)
    {
        printf("this interval does not enclose any
root !!\n");
        return ;
    }
    printf("iteration \t   a \t\t   b \t\t   c \t\t
f(c) \t\terror\n");

    while((b-a)>=epsilon)
    {
        i++;
        error=c;
        c=(a+b)/2;
        error=fabs(error-c);
        if((*p)(c) == 0.0)
          break;
        else if((*p)(c) * (*p)(a) <0)
         b=c;
        else
         a=c;
printf("%2d\t\t%4.7f\t%4.7f\t%4.7f\t%4.7f\t%4.6f\n",i,
a,b,c,(*p)(c),error);
    }
    printf("the value of approximated root is
%f\n",c);
    return ;

}
int main()
{
    float y=f(7);
```

```
     float py;
     for(int i=-7;i<=7;i++)
     {
          py=y;
          y=f((float)i);
          if(py*y<0)
          {
                printf("int this range : %d
%d\n",i-1,i);
                bisec_method(f,(float)(i-1),(float)(i));
          }
    }
     return 0;
}
```

Output:

```
int this range : -7  -6
iteration          a              b              c              f(c)           error
 1             -7.0000000     -6.5000000     -6.5000000     -2.5681982     0.500000
 2             -7.0000000     -6.7500000     -6.7500000     -0.5982352     0.250000
 3             -6.8750000     -6.7500000     -6.8750000     0.6212862      0.125000
 4             -6.8750000     -6.8125000     -6.8125000     -0.0147092     0.062500
 5             -6.8437500     -6.8125000     -6.8437500     0.2960714      0.031250
 6             -6.8281250     -6.8125000     -6.8281250     0.1389651      0.015625
 7             -6.8203125     -6.8125000     -6.8203125     0.0617094      0.007813
 8             -6.8164063     -6.8125000     -6.8164063     0.0233967      0.003906
 9             -6.8144531     -6.8125000     -6.8144531     0.0043181      0.001953
10             -6.8144531     -6.8134766     -6.8134766     -0.0052020     0.000977
11             -6.8144531     -6.8139648     -6.8139648     -0.0004436     0.000488
12             -6.8142090     -6.8139648     -6.8142090     0.0019368      0.000244
13             -6.8140869     -6.8139648     -6.8140869     0.0007465      0.000122
14             -6.8140259     -6.8139648     -6.8140259     0.0001515      0.000061
15             -6.8140259     -6.8139954     -6.8139954     -0.0001461     0.000031
16             -6.8140106     -6.8139954     -6.8140106     0.0000027      0.000015
17             -6.8140106     -6.8140030     -6.8140030     -0.0000717     0.000008
the value of approximated root is -6.814003
int this range : -5  -4
iteration          a              b              c              f(c)           error
 1             -5.0000000     -4.5000000     -4.5000000     16.8679943     0.500000
 2             -4.7500000     -4.5000000     -4.7500000     -130.2332153   0.250000
 3             -4.7500000     -4.6250000     -4.6250000     48.7895012     0.125000
 4             -4.7500000     -4.6875000     -4.6875000     184.2974701    0.062500
 5             -4.7187500     -4.6875000     -4.7187500     -745.8128662   0.031250
 6             -4.7187500     -4.7031250     -4.7031250     503.6641235    0.015625
 7             -4.7187500     -4.7109375     -4.7109375     3241.6064453   0.007813
 8              -4.7148438      -4.7109375      -4.7148438      -1924.6829834   0.003906
 9              -4.7128906      -4.7109375      -4.7128906      -9398.8789063   0.001953
10              -4.7128906      -4.7119141      -4.7119141      9917.5332031    0.000977
11              -4.7124023      -4.7119141      -4.7124023      -352639.9062500 0.000488
12              -4.7124023      -4.7121582      -4.7121582      20414.6406250   0.000244
13              -4.7124023      -4.7122803      -4.7122803      43344.4726563   0.000122
14              -4.7124023      -4.7123413      -4.7123413      98845.6796875   0.000061
15              -4.7124023      -4.7123718      -4.7123718      274702.3750000  0.000031
16              -4.7124023      -4.7123871      -4.7123871      2486187.7500000 0.000015
17              -4.7123947      -4.7123871      -4.7123947      -821841.9375000 0.000008
the value of approximated root is -4.712395
int this range : -4  -3
iteration          a               b               c               f(c)          error
 1              -4.0000000      -3.5000000      -3.5000000      -2.6889503      0.500000
 2              -4.0000000      -3.7500000      -3.7500000      -1.3879343      0.250000
 3              -4.0000000      -3.8750000      -3.8750000      -0.5083439      0.125000
 4              -3.9375000      -3.8750000      -3.9375000      0.0211419       0.062500
 5              -3.9375000      -3.9062500      -3.9062500      -0.2525174      0.031250
 6              -3.9375000      -3.9218750      -3.9218750      -0.1180483      0.015625
 7              -3.9375000      -3.9296875      -3.9296875      -0.0490609      0.007813
 8              -3.9375000      -3.9335938      -3.9335938      -0.0141137      0.003906
 9              -3.9355469      -3.9335938      -3.9355469      0.0034753       0.001953
10              -3.9355469      -3.9345703      -3.9345703      -0.0053289      0.000977
11              -3.9355469      -3.9350586      -3.9350586      -0.0009292      0.000488
12              -3.9353027      -3.9350586      -3.9353027      0.0012724       0.000244
13              -3.9351807      -3.9350586      -3.9351807      0.0001714       0.000122
14              -3.9351807      -3.9351196      -3.9351196      -0.0003789      0.000061
15              -3.9351807      -3.9351501      -3.9351501      -0.0001038      0.000031
16              -3.9351654      -3.9351501      -3.9351654      0.0000338       0.000015
17              -3.9351654      -3.9351578      -3.9351578      -0.0000350      0.000008
the value of approximated root is -3.935158
int this range : 3  4
iteration          a               b               c               f(c)          error
 1              3.5000000       4.0000000       3.5000000       -2.6889503      0.500000
 2              3.7500000       4.0000000       3.7500000       -1.3879343      0.250000
 3              3.8750000       4.0000000       3.8750000       -0.5083439      0.125000
 4              3.8750000       3.9375000       3.9375000       0.0211419       0.062500
 5              3.9062500       3.9375000       3.9062500       -0.2525174      0.031250
 6              3.9218750       3.9375000       3.9218750       -0.1180483      0.015625
 7              3.9296875       3.9375000       3.9296875       -0.0490609      0.007813
 8              3.9335938       3.9375000       3.9335938       -0.0141137      0.003906
 9              3.9335938       3.9355469       3.9355469       0.0034753       0.001953
10              3.9345703       3.9355469       3.9345703       -0.0053289      0.000977
11              3.9350586       3.9355469       3.9350586       -0.0009292      0.000488
12              3.9350586       3.9353027       3.9353027       0.0012724       0.000244
13              3.9350586       3.9351807       3.9351807       0.0001714       0.000122
14              3.9351196       3.9351807       3.9351196       -0.0003789      0.000061
15              3.9351501       3.9351807       3.9351501       -0.0001038      0.000031
16              3.9351501       3.9351654       3.9351654       0.0000338       0.000015
17              3.9351578       3.9351654       3.9351578       -0.0000350      0.000008
the value of approximated root is 3.935158
int this range : 4  5
iteration          a               b               c               f(c)          error
 1              4.5000000       5.0000000       4.5000000       16.8679943      0.500000
 2              4.5000000       4.7500000       4.7500000       -130.2332153    0.250000
 3              4.6250000       4.7500000       4.6250000       48.7895012      0.125000
 4              4.6875000       4.7500000       4.6875000       184.2974701     0.062500
 5              4.6875000       4.7187500       4.7187500       -745.8128662    0.031250
 6              4.7031250       4.7187500       4.7031250       503.6641235     0.015625
 7              4.7109375       4.7187500       4.7109375       3241.6064453    0.007813
 8              4.7109375       4.7148438       4.7148438       -1924.6829834   0.003906
 9              4.7109375       4.7128906       4.7128906       -9398.8789063   0.001953
10              4.7119141       4.7128906       4.7119141       9917.5332031    0.000977
11              4.7119141       4.7124023       4.7124023       -352639.9062500 0.000488
12              4.7121582       4.7124023       4.7121582       20414.6406250   0.000244
13              4.7122803       4.7124023       4.7122803       43344.4726563   0.000122
14              4.7123413       4.7124023       4.7123413       98845.6796875   0.000061
15              4.7123718       4.7124023       4.7123718       274702.3750000  0.000031
16              4.7123871       4.7124023       4.7123871       2486187.7500000 0.000015
17              4.7123871       4.7123947       4.7123947       -821841.9375000 0.000008
the value of approximated root is 4.712395
int this range : 6  7
iteration         a               b               c               f(c)          error
 1              6.5000000       7.0000000       6.5000000       -2.5681982      0.500000
 2              6.7500000       7.0000000       6.7500000       -0.5982352      0.250000
 3              6.7500000       6.8750000       6.8750000       0.6212862       0.125000
 4              6.8125000       6.8750000       6.8125000       -0.0147092      0.062500
 5              6.8125000       6.8437500       6.8437500       0.2960714       0.031250
 6              6.8125000       6.8281250       6.8281250       0.1389651       0.015625
 7              6.8125000       6.8203125       6.8203125       0.0617094       0.007813
 8              6.8125000       6.8164063       6.8164063       0.0233967       0.003906
 9              6.8125000       6.8144531       6.8144531       0.0043181       0.001953
10              6.8134766       6.8144531       6.8134766       -0.0052020      0.000977
11              6.8139648       6.8144531       6.8139648       -0.0004436      0.000488
12              6.8139648       6.8142090       6.8142090       0.0019368       0.000244
13              6.8139648       6.8140869       6.8140869       0.0007465       0.000122
14              6.8139648       6.8140259       6.8140259       0.0001515       0.000061
15              6.8139954       6.8140259       6.8139954       -0.0001461      0.000031
16              6.8139954       6.8140106       6.8140106       0.0000027       0.000015
17              6.8140030       6.8140106       6.8140030       -0.0000717      0.000008
the value of approximated root is 6.814003
```

Numerical Methods

Practical - 3

Objective: There are three real roots of the equation x3 – 2.5x2 – 2.46x + 3.96 = 0 in the domain [-4, +4]. Write a program to first find out the disjoint subintervals in the given domain that cover the roots. Hence find the roots by Newton-Raphson method.

Code:

```
#include<stdio.h>
#include<math.h>
 float fun(float x)
 {
    return pow(x,3)-(2.5)*pow(x,2)-(2.24)*x+3;
 }
 void bisection(float g1,float g2,float e)
 {


     float y1=fun(g1);
     float y2=fun(g2);

   if(y1*y2>0)
   {
    printf("no roots lie \n");
    return ;
   }
   else if(y1==0)
{
    printf("%f is a root\n",g1);
    return ;
}
  else if(y2==0)
{
    printf("%f is a root\n",g2);
    return ;
}
    int i=1;
    float x=0;
    while(fabs(g2-g1)>e && i<100)
    {
        x=(g1+g2)/2;
        float y=fun(x);
        i++;

       if(y*y2<0)
       {
         g1=x;

         g2=g2;
       }
       else if(y1*y<0)
       {
        g1=g1;
        g2=x;
       }
       else if(y==0)
       {
         printf("%f is a root\n",x);
            return ;
       }

    }
    printf("approximated root is %f\n",x);

 }
```

```
int main()
{
    float y=fun(-4);
    float py;
    for(int i=-4;i<=4;i++)
    {
        py=y;
        y=fun(i);
        if(py*y<0)
        {
            printf("int this range : %d  %d\n",i-1,i);
            bisection(i-1,i,0.001);
        }
    }
    return 0;
}
```

Output:

```
int this range : -2  -1
approximated root is -1.243164
int this range : 0  1
approximated root is 0.827148
int this range : 2  3
approximated root is 2.915039
```