C Programming

Day 1

Program 1:- WAP to print "Hello world on screen"

```c
#include <stdio.h>
int main() {
   printf("Hello, World!");
   return 0;
}
```

Output : 10 + 20 = 30

Program 2:- WAP for addition of two number by taking static input

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

    int number1=10, number2=20, sum;

    sum = number1 + number2;

    printf("%d + %d = %d", number1, number2, sum);
    return 0;
}
```

Output : 10 + 20 = 30

Program 3:- WAP for addition of two number using dynamic input.

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

    int number1, number2, sum;

    printf("Enter two integers: ");
    scanf("%d %d", &number1, &number2);

    sum = number1 + number2;

    printf("%d + %d = %d", number1, number2, sum);
    return 0;
```

```
}
```

```
Output : Enter two integers: 10  11

10 + 11 = 21
```

Program 4 :- WAP to Print the date and time in the default format

```
#include<stdio.h>
#include<conio.h>
#include<time.h>

int main()
{
    time_t tm;
    time(&tm);
    printf("Current Date/Time = %s", ctime(&tm));
    getch();
    return 0;
}
```

```
Output : Current Date/Time = Wed May 22 17:11:13 2024
```

Program 5 :- WAP to rake input from user and enter their name on screen.

```
#include<stdio.h>

void main()
{
    char ch[20];

    printf("Enter name : ");
    scanf("%s",ch);

    printf("Entered name is : %s\n",ch);
}
```

```
Output : Enter name : Sahil

Entered name is : Sahil
```

Program 6 :- WAP to check given number is even or odd.

```c
#include <stdio.h>

int main() {
int number;

printf("Enter a number: ");
scanf("%d", &number);

    if (number % 2 == 0)
    {
        printf("%d is an even number.\n", number);
    }
    else
    {
        printf("%d is an odd number.\n", number);
    }

    return 0;
}
```

```
Output : Enter a number: 125

125 is an odd number.
```

Program 7 :- WAP which gives ASCII value of given character.

```c
#include<stdio.h>

void main()
{
    char ch;

    printf("Enter the character : ");
    scanf("%c",&ch);

    printf("ASCII value of %c = %d \n ",ch,ch);
}
```

```
Output : Enter the character : G

ASCII value of G = 71
```

Program 8 :- WAP which gives Quotient and Remainder.

```c
#include<stdio.h>

void main()
{
    int dv,dd,q,r;

    printf("Enter the number : ");
    scanf("%d",&dd);

    printf("Enter the divisor : ");
    scanf("%d",&dv);

    q = dd / dv;
    r = dd % dv;

    printf("Quotient is %d \n",q);
    printf("Remainder is %d \n",r);

}
```

```
Output : Enter the number : 120

Enter the divisor : 4

Quotient is 30

Remainder is 0
```

Program 9 :- WAP to Reverse the given number

```c
#include<stdio.h>

 int main()
{
    int n, reverse=0, rem;

    printf("Enter a number: ");
    scanf("%d", &n);

    while(n!=0)
    {
        rem=n%10;
        reverse=reverse*10+rem;
        n/=10;
    }
    printf("Reversed Number: %d",reverse);
    return 0;
}
```

```
Output : Enter a number: 125634
Reversed Number: 436521
```

Program 10 :- WAP to Find the Size of data types

```c
#include<stdio.h>
int main()
{
    int intType;
    float floatType;
    double doubleType;
    char charType;
    unsigned u;

    printf("Size of int: %zu bytes\n", sizeof(intType));
    printf("Size of float: %zu bytes\n", sizeof(floatType));
    printf("Size of double: %zu bytes\n", sizeof(doubleType));
    printf("Size of char: %zu byte\n", sizeof(charType));
    printf("Size of unsigned: %zu byte\n", sizeof(u));

    return 0;
}
```

```
Output : Size of int: 4 bytes
Size of float: 4 bytes
Size of double: 8 bytes
Size of char: 1 byte
Size of unsigned: 4 byte
```

Day 2

Program 11 :- Pattern printing

```c
#include<stdio.h>

void main()
{
    int i, j, ns=0;

    system ("cls");

    printf("Enter the number rows for star \n");
    scanf("%d",&ns);

    for(i=1; i<= ns; i++){
        for(j=1; j<=i; j++){
            printf("*");
        }
        printf("\n");
    }
}
```

```
Output : Enter the number rows for star : 5
*
**
***
****
*****
```

Program 12 :- Matrix program

```c
#include<stdio.h>
#include<conio.h>
#include<math.h>

void main()
{
    system("cls");
    int i, j, r,c,n[2][2];

    printf("Maximum values [2][2]\n");

    for(i=0; i<2; i++){
        for(j=0; j<2; j++){
            scanf("%d",&n[i][j]);
        }
```

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

    printf("The matrix is :: \n");
    for(i=0; i<2;i++){
        for(j=0; j<2; j++){
            printf("%d",n[i][j]);
        }
        printf("\n");
    }
}
```

```
Output : Maximum values [2][2]
3
2

3
4

The matrix is ::
3     2
3     4
```

Program 13 :- Matrix multiplication

```
#include<stdio.h>
#include<stdlib.h>
void main()
{
    int a[2][2],b[2][2],res[2][2],i,j,k;
    system("cls");
    printf("enter the first matrix elements : \n");
    for(i=0;i<2;i++)
    {
        for(j=0;j<2;j++)
        {
            scanf("%d",&a[i][j]);
        }
    }
    printf("enter the second matrix elements : \n");
    for(i=0;i<2;i++)
        {
            for(j=0;j<2;j++)
            {
                scanf("%d",&b[i][j]);
            }
```

```
        }
    printf("Multiply of the matrix : \n");
    for(i=0;i<2;i++)
    {
        for(j=0;j<2;j++)
        {
            res[i][j]=0;
            for(k=0;k<2;k++)
            {
                res[i][j]+=a[i][k]*b[k][j];
            }
        printf("%d  ",res[i][j]);
        }
    printf("\n");
    }
}
```

```
Output : enter the first matrix elements :
1
3
4
6
enter the second matrix elements :
2
3
5
6
Multiply of the matrix :
17  21
38  48
```

Program 14 :- WAP for Fibonacci series

```
#include<stdio.h>

int main()
{
    int n = 10;
    int a = 0, b = 1;

    printf("%d %d\n",a,b);

    int nextTerm;

    for(int i = 2; i < n; i++){
        nextTerm = a + b;
```

```
        a = b;
        b = nextTerm;

        printf("%d, ",nextTerm);
    }

    return 0;
}
```

```
Output : 0 1
1, 2, 3, 5, 8, 13, 21, 34,
```

Program 15 :- WAP to find factorial of number.

```
#include <stdio.h>
int  main()
{
  int i, f = 1, num;

  printf("Input the number : ");
  scanf("%d", &num);

  for(i = 1; i <= num; i++)
     {
         f = f * i;
     }

  printf("The Factorial of %d is: %d\n", num, f);
   return 0;
}
```

```
Output : Input the number : 4
The Factorial of 4 is: 24
```

Task on Array

Program 16 :- WAP which takes 20 values as input from user and store values in array

```
#include<stdio.h>
void main()
{
    int a[20],i;
```

```
    printf(" Enter the elements of the array of 20 : \n ");
    for(i=0;i<20;i++)
    {
        scanf("%d",&a[i]);
    }
}
```

```
Output : Enter the elements of the array of 20 :
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
```

Program 17 :- Print all elements in a[20] array.

```
#include<stdio.h>
void main()
{
    int a[20],i;
    printf(" Enter the elements of the array of 20 : \n ");
    for(i=0;i<20;i++)
    {
        scanf("%d",&a[i]);
    }
    printf("\n printing the elements of the array with index values :\n");
    for(i=0;i<20;i++)
    {
        printf("a[%d] = %d\n",i,a[i]);
    }
}
```

```
Output : Enter the elements of the array of 20 :
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20

printing the elements of the array with index values :
a[0] = 1
a[1] = 2
a[2] = 3
a[3] = 4
a[4] = 5
a[5] = 6
a[6] = 7
a[7] = 8
a[8] = 9
a[9] = 10
a[10] = 11
a[11] = 12
a[12] = 13
a[13] = 14
a[14] = 15
a[15] = 16
a[16] = 17
a[17] = 18
a[18] = 19
a[19] = 20
```

Program 18 :- Delete particular elements in a[10] array

```
#include<stdio.h>
void main()
{
    int a[10],i,j;
    printf(" Enter the elements of the array of 10 : \n ");
    for(i=0;i<10;i++)
    {
        scanf("%d",&a[i]);
    }
    printf("\n Enter the index of value which is to be deleted : ");
    scanf("%d",&j);
    a[j]=0;
    printf("\n printing the elements of the array after deleting the element
:\n");
    for(i=0;i<10;i++)
    {
        printf("%d  ",a[i]);
    }
}
```

```
Output : Enter the elements of the array of 10 :
 1
2
3
4
5
6
7
8
9
10

 Enter the index of value which is to be deleted : 5

 printing the elements of the array after deleting the element :
1  2  3  4  5  0  7  8  9  10
```

Program 19 :- WAP to find if there is any duplicate element in a[10]

```
#include<stdio.h>
void main()
{
    int a[20],i,j;
    printf(" Enter the elements of the array of 10 : \n ");
    for(i=0;i<10;i++)
```

```
    {
        scanf("%d",&a[i]);
    }
    printf("\n Printing index values of duplicate elements : \n");
    for (i=0;i<10;i++)
    {
        for(j=i+1;j<10;j++)
        {
            if(a[i]==a[j])
            {
                printf("%d and %d \n",i,j);
            }
        }
    }
}
```

```
Output : Enter the elements of the array of 10 :
 1
2
3
4
5
6
7
8
9
1

 Printing index values of duplicate elements :
0 and 9
```

Program 20 :- WAP to search elements in array.

```
#include<stdio.h>
void main()
{
    int a[20],i,num,flag=0;
    printf(" Enter the elements of the array of 20 : \n ");
    for(i=0;i<20;i++)
    {
        scanf("%d",&a[i]);
    }
    printf("\n Enter the element which is to be searched :\n");
    scanf("%d",&num);
    for(i=0;i<20;i++)
    {
```

```
        if(a[i]==num)
        {
            printf(" Given number is found at index value : %d \n",i);
            flag++;
        }
    }
    if(flag==0)
    {
        printf(" Element NOT FOUND ");
    }
}
```

```
Output : Enter the elements of the array of 20 :
 1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20

 Enter the element which is to be searched : 5
 Given number is found at index value : 4
```

Day 3

Program 21 :- file handling concept

```c
#include<stdio.h>
#include <conio.h>
#include <stdlib.h>
void main ()
{
    int n;
    FILE *fptr;
    fptr = fopen ("D:\\Wipro domain training\\Day 9\\testtext.txt", "r");
    if (fptr == NULL)
    {
        printf ("Error!!!!!");
        exit(0);
    }
    printf ("Enter number::");
    scanf ("%d", &n);

    fprintf (fptr, "%d", n);
    fclose (fptr);
}
```

Program 22 :- structure concept

```c
#include <stdio.h>
#include <string.h>
struct emp
{
    char name[30];
    int age;
    float salary;
    }p1;
void main()
{
     strcpy (p1.name, "myname");
     p1.age = 27;
     p1.salary = 10000;
     printf("Name::%s \n", p1.name);
     printf("age::%d \n", p1.age);
     printf("Salary::%.3f \n", p1.salary);
}
```

Output : Name::myname

```
age::27
Salary::10000.000
```

Program 23::- Create an employee record by using struct (name, department, age) and redirect the output to .txt file using file handling

```c
#include <stdio.h>
#include <stdlib.h>
// Define a struct for employee record
struct Employee {
    char name[50];
    char department[50];
    int age;
};
int main() {
    struct Employee emp;

    printf("Enter employee name: ");
    scanf("%s",&emp.name);

    printf("Enter employee department: ");
    scanf("%s",&emp.department);

    printf("Enter employee age: ");
    scanf("%d", &emp.age);

    FILE *fptr;
    fptr = fopen ("D:\\Wipro domain training\\Day 9\\program13.txt", "a");
    if (fptr == NULL)
    {
        printf ("Error!!!!!");
        exit(0);
    }
    fprintf (fptr, "employee name : %s \n", emp.name);
    fprintf (fptr, "employee department : %s\n", emp.department);
    fprintf (fptr, " employee age : %d\n", emp.age);
    return 0;
}
```

```
Output : Enter employee name: Ganesh
Enter employee department: Support
Enter employee age: 23
Text file : employee name : Ganesh
employee department : Support
 employee age : 23
```

Day 4

Program 24 :- WAP which takes year and month from user and gives dates in that month

```c
#include <stdio.h>

int isLeapYear(int year) {
    return ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0));
}


int getDaysInMonth(int month, int year) {
    switch (month) {
        case 4: case 6: case 9: case 11:
            return 30;
        case 2:
            return isLeapYear(year) ? 29 : 28;
        default:
            return 31;
    }
}

int main() {
    int year, month;

    printf("Enter the year: ");
    scanf("%d", &year);
    printf("Enter the month number (1-12): ");
    scanf("%d", &month);

    if (month < 1 || month > 12) {
        printf("Invalid month number!\n");
        return 1;
    }

    printf("Dates for %d/%d:\n", month, year);
    int daysInMonth = getDaysInMonth(month, year);
    for (int day = 1; day <= daysInMonth; ++day) {
        printf("%d/%d/%d\n", month, day, year);
    }

    return 0;
}
```

```
Output : Enter the year: 2024
Enter the month number (1-12): 5
Dates for 5/2024:
```

```
5/1/2024        5/2/2024        5/3/2024        5/4/2024        5/5/2024
5/6/2024        5/7/2024        5/8/2024        5/9/2024        5/10/2024
5/11/2024       5/12/2024       5/13/2024       5/14/2024       5/15/2024
5/16/2024       5/17/2024       5/18/2024       5/19/2024       5/20/2024
5/21/2024       5/22/2024       5/23/2024       5/24/2024       5/25/2024
5/26/2024       5/27/2024       5/28/2024        5/29/2024       5/30/2024
5/31/2024
```

Program 25 :- WAP which takes month andyear from user and gives calender of that month.

```c
#include <stdio.h>

int dayOfWeek(int d, int m, int y) {
    static int t[] = {0, 3, 2, 5, 0, 3, 5, 1, 4, 6, 2, 4};
    y -= m < 3;
    return (y + y/4 - y/100 + y/400 + t[m-1] + d) % 7;
}

void printCalendar(int month, int year) {
    int daysInMonth, i, currentDay;
    int days[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
    if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0))
        days[1] = 29;

    printf("  ****  Calendar - %d/%d  ****\n", month, year);
    printf("  Sun  Mon  Tue  Wed  Thu  Fri  Sat\n");

    currentDay = dayOfWeek(1, month, year);

    for (i = 0; i < currentDay; i++)
        printf("     ");
    for (i = 1; i <= days[month-1]; i++) {
        printf("%5d", i);
        if (++currentDay > 6) {
            currentDay = 0;
           printf("\n");
        }
    }
    if (currentDay != 0)
        printf("\n");
}

int main() {
    int month, year;
    printf("Enter month and year (MM YYYY): ");
    scanf("%d %d", &month, &year);
    printCalendar(month, year);
```

```
    return 0;
}
```

```
Output : Enter month and year (MM YYYY): 5
2024
  ****  Calendar - 5/2024  ****
  Sun  Mon  Tue  Wed  Thu  Fri  Sat
                    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
```

Day 5

Program 26 :- pointer concept program

```c
#include <stdio.h>

int main() {
    // Write C code here
    int *p;
    int n;
    n=0x18;
    p = &n;

    *p = *p + 4;
    printf("%d\n",n);

    n = *p+4;
    printf("%d\n",n);

    return 0;
}
```

```
Output :
28
32
```

Program 27 :- pointer address storing concept

```c
#include<stdio.h>
#include<conio.h>
void main()
{
    int x,y;
    int *ptr;
    x=10;
    ptr = &x;
    y= *ptr;
    printf ("%d : (x) is stored in location :: %u \n", x, &x);
    printf ("%d : (*&x) is stored in location :: %u \n", *&x, &x);
    printf ("%d : (*ptr) is stored in location :: %u \n", *ptr, ptr);
    printf ("%d : (y) is stored in location :: %u \n", y, &*ptr);
    printf ("%u : (ptr = &x) is stored in location :: %u \n", ptr, &ptr);
    printf ("%d : (y) is stored in location :: %u \n", y, &y);

    getch();
}
```

```
Output :
10 : (x) is stored in location :: 6422300
10 : (*&x) is stored in location :: 6422300
10 : (*ptr) is stored in location :: 6422300
10 : (y) is stored in location :: 6422300
6422300 : (ptr = &x) is stored in location :: 6422292
10 : (y) is stored in location :: 6422296
```

Program 28 :- WAP which shows sizeof data type.

```c
#include <stdio.h>
#include <conio.h>

void main()
{
   printf ("No. of Bytes occupied by int is %d \n", sizeof(int));
   printf ("No. of Bytes occupied by float is %d \n", sizeof(float));
   printf ("No. of Bytes occupied by double is %d \n", sizeof(double));
   printf ("No. of Bytes occupied by char is %d \n", sizeof(char));

   getch();

}
```

```
Output :
No. of Bytes occupied by int is 4
No. of Bytes occupied by float is 4
No. of Bytes occupied by double is 8
No. of Bytes occupied by char is 1
```

Program 29 :- Concept of Call by value and Call by reference using swap number program

```c
#include <stdio.h>

void main()
{
    int a,b;
    a=5, b=20;
    swap (a,b);
    swap1 (&a, &b);
    printf ("\n Swap Fun:  (call by value)  \n a = %d , b = %d ", a,b);
    printf ("\n Swap1 Fun:  (call by Ref)  \n a = %d , b = %d ", a,b);
}

void swap (int x, int y)
```

```
{
    int tmp;
    tmp = x;
    x=y;
    y=tmp;
}
void swap1 (int *x1, int *y1)
{
    int tmp1;
    tmp1 = *x1;
    *x1=*y1;
    *y1=tmp1;
}
```

```
Output :

Swap Fun:  (call by value)
 a = 20 , b = 5

 Swap1 Fun:  (call by Ref)
 a = 20 , b = 5
```

Program 30 :- WAP which shows average of an array elements.

```
#include <stdio.h>

float avg (int arr[], int size);
void main ()
{
    int x[100], k, n;
    printf("\n Enter the array size :\n");
    scanf ("%d",&n);
    printf("\n Enter the array elements :\n");
    for (k=0;k<n;k++)
    {
      scanf("%d", &x[k]);
     }
     printf("\n Average is : %f", avg (x,n));
}
float avg (int arr[], int size)
{
    int *p,i,sum=0;
    p=arr;
    for (i=0;i<size;i++)
    {
        sum = sum + *(p+i);
    }
```

```
    return (float) sum/size;
}
```

```
Output :
Enter the array size :5

Enter the array elements :
10
20
30
50
60

Average is : 34.000000
```

Program 31 :- WAP for Bubble sort

```
#include <stdio.h>

void bubble_sort(int arr[], int n) {
  int i, j;
  for (i = 0; i < n - 1; i++) {
    for (j = 0; j < n - i - 1; j++) {
      if (arr[j] > arr[j + 1]) {
        int temp = arr[j];
        arr[j] = arr[j + 1];
        arr[j + 1] = temp;
      }
    }
  }
}

int main() {
  int arr[] = { 22, 11, 90,64, 34, 25, 12,1};
  int n = sizeof(arr) / sizeof(arr[0]);
  bubble_sort(arr, n);
  printf("Sorted array: ");
  for (int i = 0; i < n; i++) {
    printf("%d ", arr[i]);
  }

  return 0;
}
```

```
Output :
Sorted array: 1 11 12 22 25 34 64 90
```

Day 6

Program 32 :- WAP which take input from user and show it on screen

```c
#include <stdio.h>

void main()
{

    int arr[5];
    int i,j;
    printf("Enter the Array Elements::\n");
    for (i=0;i<5;i++)
    {
        scanf("%d",&arr[i]);
    }
    for (j=0;j<5;j++)
    {
        printf("a[%d] is : : %d\n",j,arr[j]);
    }
    getch();
}
```

```
Output : Enter the Array Elements::
10
20
30
40
50

a[0] is : : 10
a[1] is : : 20
a[2] is : : 30
a[3] is : : 40
a[4] is : : 50
```

Program 33 :- WAP to delete the array index elements from array.

```c
#include <stdio.h>

void main()
{

    int arr[5];
    int i,j,n,counter=0;
    printf("Enter the Array Elements::\n");
    for (i=0;i<5;i++)
```

```
    {
        scanf("%d",&arr[i]);
    }
    for (j=0;j<5;j++)
    {
        printf("a[%d] is : : %d\n",j,arr[j]);
    }

    printf("Enter the Array Index you want to delete::\n");
    scanf("%d",&n);
    arr[n] = 0;

    printf("Array Elements after Deletion::\n");
    for (j=0;j<5;j++)
    {
        printf("a[%d] is : : %d\n",j,arr[j]);
    }

    for (i=0;i<5;i++)
    {
        if (arr[i]== 0)
            counter = counter +1;
    }
    printf("Total Empty spaces available :: %d\n", counter);
    getch();
}
```

```
Output : Enter the Array Elements::
10
20
30
40
50

a[0] is : : 10
a[1] is : : 20
a[2] is : : 30
a[3] is : : 40
a[4] is : : 50

Enter the Array Index you want to delete::
2

Array Elements after Deletion::
a[0] is : : 10
a[1] is : : 20
a[2] is : : 0
a[3] is : : 40
```

```
a[4] is : : 50

Total Empty spaces available :: 1
```

Program 34 :- WAP to delete the array index elements from array after deleting program ask to continue or not.

```c
#include <stdio.h>

int main()
{

    int arr[5];
    int i,j,n,ch;
    printf("Enter the Array Elements::\n");
    for (i=0;i<5;i++)
    {
        scanf("%d",&arr[i]);
    }
    for (j=0;j<5;j++)
    {
        printf("a[%d] is : : %d\n",j,arr[j]);
    }

    del1:

    printf("Enter the Array Index you want to delete::\n");
    scanf("%d",&n);
    arr[n] = 0;

    printf("Array Elements after Deletion::\n");
    for (j=0;j<5;j++)
    {
        printf("a[%d] is : : %d\n",j,arr[j]);
    }
    int counter=0;
    for (i=0;i<5;i++)
    {
        if (arr[i]== 0)
            counter = counter +1;
    }
    printf("Total Empty spaces available :: %d\n", counter);
    printf("Do you want to continue if yes press 1 or o if no \n");
    scanf("%d",&ch);

    if(ch==1)
    {
        goto del1;
```

```
    }
    else{return 0;}
    return 0;

}
```

```
Output :
Enter the Array Elements::
10
20
30
40
50

a[0] is : : 10
a[1] is : : 20
a[2] is : : 30
a[3] is : : 40
a[4] is : : 50

Enter the Array Index you want to delete::
2
Array Elements after Deletion::
a[0] is : : 10
a[1] is : : 20
a[2] is : : 0
a[3] is : : 40
a[4] is : : 50

Total Empty spaces available :: 1
Do you want to continue if yes press 1 or o if no
1

Enter the Array Index you want to delete::
1

Array Elements after Deletion::
a[0] is : : 10
a[1] is : : 0
a[2] is : : 0
a[3] is : : 40
a[4] is : : 50

Total Empty spaces available :: 2

Do you want to continue if yes press 1 or o if no
0
```

Program 34 :- WAP for binary search.

```c
#include <stdio.h>

int binarySearch(int array[], int x, int low, int high) {

  while (low <= high) {
    int mid = low + (high - low) / 2;

    if (array[mid] == x)
      return mid;

    if (array[mid] < x)
      low = mid + 1;

    else
      high = mid - 1;
  }

  return -1;
}

int main(void) {
  int array[] = {3, 4, 5, 6, 7, 8, 9};
  int n = sizeof(array) / sizeof(array[0]);
  int x = 4;
  int result = binarySearch(array, x, 0, n - 1);
  if (result == -1)
    printf("Not found");
  else
    printf("Element is found at index %d", result);
  return 0;
}
```

Output :

Element is found at index 1

Program 35 :- WAP for Stack in data structure

```c
#include <stdio.h>
#include <stdlib.h>

#define MAX 10
int count = 0;
```

```c
// Creating a stack
struct stack {
  int items[MAX];
  int top;
};
typedef struct stack st;

void createEmptyStack(st *s) {
  s->top = -1;
}

// Check if the stack is full
int isfull(st *s) {
  if (s->top == MAX - 1)
    return 1;
  else
    return 0;
}

// Check if the stack is empty
int isempty(st *s) {
  if (s->top == -1)
    return 1;
  else
    return 0;
}

// Add elements into stack
void push(st *s, int newitem) {
  if (isfull(s)) {
    printf("STACK FULL");
  } else {
    s->top++;
    s->items[s->top] = newitem;
  }
  count++;
}

// Remove element from stack
void pop(st *s) {
  if (isempty(s)) {
    printf("\n STACK EMPTY \n");
  } else {
    printf("Item popped= %d", s->items[s->top]);
    s->top--;
  }
  count--;
  printf("\n");
```

```
}

// Print elements of stack
void printStack(st *s) {
  printf("Stack: ");
  for (int i = 0; i < count; i++) {
    printf("%d ", s->items[i]);
  }
  printf("\n");
}

// Driver code
int main() {
  int ch;
  st *s = (st *)malloc(sizeof(st));

  createEmptyStack(s);

  push(s, 1);
  push(s, 2);
  push(s, 3);
  push(s, 4);

  printStack(s);

  pop(s);

  printf("\nAfter popping out\n");
  printStack(s);
}
```

```
Output :
Stack: 1 2 3 4
Item popped= 4

After popping out
Stack: 1 2 3
```

Program 36 :- WAP for Stack in data structure

```
#include <stdio.h>
#define SIZE 5

void enQueue(int);
```

```c
void deQueue();
void display();

int items[SIZE], front = -1, rear = -1;

int main() {
  //deQueue is not possible on empty queue
  deQueue();

  //enQueue 5 elements
  enQueue(1);
  enQueue(2);
  enQueue(3);
  enQueue(4);
  enQueue(5);

  // 6th element can't be added to because the queue is full
  enQueue(6);

  display();

  //deQueue removes element entered first i.e. 1
  deQueue();

  //Now we have just 4 elements
  display();

  return 0;
}

void enQueue(int value) {
  if (rear == SIZE - 1)
    printf("\nQueue is Full!!");
  else {
    if (front == -1)
      front = 0;
    rear++;
    items[rear] = value;
    printf("\nInserted -> %d", value);
  }
}

void deQueue() {
  if (front == -1)
    printf("\nQueue is Empty!!");
  else {
    printf("\nDeleted : %d", items[front]);
    front++;
```

```
    if (front > rear)
      front = rear = -1;
  }
}

// Function to print the queue
void display() {
  if (rear == -1)
    printf("\nQueue is Empty!!!");
  else {
    int i;
    printf("\nQueue elements are:\n");
    for (i = front; i <= rear; i++)
      printf("%d  ", items[i]);
  }
  printf("\n");
}
```

```
Output :
Queue is Empty!!
Inserted -> 1
Inserted -> 2
Inserted -> 3
Inserted -> 4
Inserted -> 5
Queue is Full!!

Queue elements are:
1 2 3 4 5

Deleted : 1
Queue elements are:
2 3 4 5
```

Program 37::- WAP to implement singly Linked list data structure

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <conio.h>

struct node{
    int data;
    struct node *link;
    };
```

```
void insert(struct node **head, int data)
{
    struct node *newnode = (struct node *) malloc (sizeof (struct node));
    newnode->data = data;
    newnode->link = *head;
    *head = newnode ;
}
void display (struct node *Node)
{
    while (Node != NULL)
    {
        printf ("%d\t", Node->data);
        Node = Node->link;
    }
    printf("\n");
}
main()
{
    struct node *head = NULL;
    struct node *node2 = NULL;
    struct node *node3 = NULL;

    head = (struct node *) malloc (sizeof (struct node));
    node2 = (struct node *) malloc (sizeof (struct node));
    node3 = (struct node *) malloc (sizeof (struct node));

    head->data = 9;
    head->link = node2;
    node2->data = 10;
    node2->link = node3;
    node3->data = 11;
    node3->link = NULL;

    printf("Elements are:: \n");
    display (head);
    insert(&head, 12);
    printf ("After inserting ::\n");
    display (head);
    insert(&head, 13);
    printf ("After inserting ::\n");
    display (head);

getch();

}
```

```
Output : Elements are::
9    10    11
```

```
After inserting ::
12    9    10    11
After inserting ::
13    12    9    10    11
```

Program 38 ::- write singly linked list code with all function

```c
#include<stdio.h>
#include<stdlib.h>

struct node
{
    int data;           // 4
    struct node *next;  // 8
};

typedef struct node NODE;
typedef struct node * PNODE;
typedef struct node ** PPNODE;

void InsertFirst(PPNODE head, int no)
{
    // Allocate memory for node (dynamically)
    // Initialise that node

    // Check whether LL is empty or not
    // If LL is empty then new node is the first node so update its address in
first pointer through head

    // If LL is not empty then store the address of first node in the next
pointer of our new node
    // update the first pointer thruogh head
    PNODE newn = NULL;

    newn = (PNODE)malloc(sizeof(NODE));     // newn = (struct node
*)malloc(12);

    newn->data = no;
    newn->next = NULL;

    if(*head == NULL)   // LL is empty
    {
        *head = newn;
    }
    else        // LL contains atleast one node
    {
        newn->next = *head;
```

```
        *head = newn;
    }
}

void InsertLast(PPNODE head, int no)
{
    // Allocate memory for node (dynamically)
    // Initialise that node

    // Check whether LL is empty or not
    // If LL is empty then new node is the first node so update its address in
first pointer through head

    // If LL is not empty then
    // travel till last node of LL
    // store address of new node in the next pointer of last node

    PNODE newn = NULL;
    PNODE temp = NULL;

    newn = (PNODE)malloc(sizeof(NODE));     // newn = (struct node
*)malloc(12);

    newn->data = no;
    newn->next = NULL;

    if(*head == NULL)   // LL is empty
    {
        *head = newn;
    }
    else        // LL contains atleast one node
    {
            // travel till last node
            // store address of new node in the next pointer of last node
          temp = *head;

          while(temp->next != NULL)
          {
            temp = temp->next;
          }
          temp->next = newn;
    }
}

void Display(PNODE head)
{
    printf("Elements from linked list are : \n");
```

```c
    while(head != NULL)
    {
        printf("| %d |-> ",head->data);
        head = head -> next;
    }
    printf("NULL\n");
}

int Count(PNODE head)
{
    int iCnt = 0;

    while(head != NULL)
    {
        iCnt++;
        head = head -> next;
    }
    return iCnt;
}

void DeleteFirst(PPNODE head)
{
    // If LL is empty then return
    // If LL contains atleast one node then
    // Store the address of second node in the first pointer through head
    // And delete the fisrt node
    PNODE temp = NULL;

    if(*head == NULL)   // LL is empty
    {
        return;
    }
    else        // LL contains atleast one node
    {
        temp = *head;
        *head = temp -> next;
        free(temp);
    }
}

void DeleteLast(PPNODE head)
{
    // If LL is empty then return
    // If LL contains one node then delete that node and return
    // If LL contains more than one node then travel till second last node and
delete last node

    PNODE temp = NULL;
```

```
    if(*head == NULL)   // LL is empty
    {
        return;
    }
    else if((*head) -> next == NULL) // LL contains one node
    {
            free(*head);
            *head = NULL;
    }
    else    // LL contains more than one node
    {
            temp = *head;
           while(temp->next->next != NULL)
           {
                temp = temp -> next;
           }

           free(temp->next);
           temp->next = NULL;
    }
}

void InsertAtPos(PPNODE head, int no, int pos)
{
    // Consider no of nodes are 4

    // If position is invalid then return directly (< 1  OR > 5)
    // If position is 1 then call insertfirst
    // If position is N+1 then call Insertlast  (position is 5)

    int size = 0, iCnt = 0;
    PNODE newn = NULL;
    PNODE temp = NULL;

    size = Count(*head);

    if((pos < 1) || (pos > (size+1)))
    {
        printf("Position is invalid\n");
        return;
    }

    if(pos == 1)
    {
        InsertFirst(head,no);
    }
    else if(pos == (size+1))
```

```
    {
        InsertLast(head,no);
    }
    else        // Logic
    {
        newn = NULL;

        newn = (PNODE)malloc(sizeof(NODE));     // newn = (struct node
*)malloc(12);

        newn->data = no;
        newn->next = NULL;

        temp = *head;

        for(iCnt = 1; iCnt < pos-1 ; iCnt++)
        {
            temp = temp -> next;
        }

        newn -> next = temp -> next;
        temp->next = newn;
    }
}

void DeleteAtPos(PPNODE head, int pos)
{
    // Consider no of nodes are 4

    // If position is invalid then return directly (< 1  OR > 4)
    // If position is 1 then call deletefirst
    // If position is N then call deletelast  (position is 4)

    int size = 0, iCnt = 0;
    PNODE temp = NULL;
    PNODE tempdelete = NULL;

    size = Count(*head);

    if((pos < 1) || (pos > (size)))
    {
        printf("Position is invalid\n");
        return;
    }
    if(pos == 1)
    {
        DeleteFirst(head);
    }
```

```
    else if(pos == (size))
    {
        DeleteLast(head);
    }
    else        // Logic
    {
        temp = *head;

        for(iCnt = 1; iCnt < pos-1 ; iCnt++)
        {
            temp = temp -> next;
        }
        tempdelete = temp->next;
        temp->next = temp->next->next;
        free(tempdelete);
    }
}
int main()
{
    int iRet = 0;
    PNODE first = NULL;

    InsertFirst(&first,101);    // call by address
    InsertFirst(&first,51);
    InsertFirst(&first,21);
    InsertFirst(&first,11);

    InsertAtPos(&first,75,3);

    DeleteAtPos(&first,3);

    Display(first);     // Call by value

    iRet = Count(first);
    printf("Number of nodes are  : %d\n",iRet);

    InsertFirst(&first,1);

    Display(first);     // Call by value

    iRet = Count(first);
    printf("Number of nodes are  : %d\n",iRet);

    InsertLast(&first,111);
    InsertLast(&first,121);

    Display(first);     // Call by value
```

```
    iRet = Count(first);
    printf("Number of nodes are  : %d\n",iRet);

    DeleteFirst(&first);
    DeleteFirst(&first);

    Display(first);     // Call by value

    iRet = Count(first);
    printf("Number of nodes are  : %d\n",iRet);

    DeleteLast(&first);
    Display(first);     // Call by value

    iRet = Count(first);
    printf("Number of nodes are  : %d\n",iRet);
    return 0;

}
```

```
Output :
Elements from linked list are :
| 11 |-> | 21 |-> | 51 |-> | 101 |-> NULL

Number of nodes are  : 4

Elements from linked list are :
| 1 |-> | 11 |-> | 21 |-> | 51 |-> | 101 |-> NULL

Number of nodes are  : 5

Elements from linked list are :
| 1 |-> | 11 |-> | 21 |-> | 51 |-> | 101 |-> | 111 |-> | 121 |-> NULL

Number of nodes are  : 7

Elements from linked list are :
| 21 |-> | 51 |-> | 101 |-> | 111 |-> | 121 |-> NULL

Number of nodes are  : 5

Elements from linked list are :
| 21 |-> | 51 |-> | 101 |-> | 111 |-> NULL

Number of nodes are  : 4
```

Program 39 ::- write doubly linked list code with all function

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <stdbool.h>
struct node {
   int data;
   int key;
   struct node *next;
   struct node *prev;
};

//this link always point to first Link
struct node *head = NULL;

//this link always point to last Link
struct node *last = NULL;
struct node *current = NULL;

//is list empty
bool isEmpty(){
   return head == NULL;
}

//display the doubly linked list
void printList(){
   struct node *ptr = head;
   while(ptr != NULL) {
      printf("(%d,%d) ",ptr->key,ptr->data);
      ptr = ptr->next;
   }
}

//insert link at the first location
void insertFirst(int key, int data){

   //create a link
   struct node *link = (struct node*) malloc(sizeof(struct node));
   link->key = key;
   link->data = data;
   if(isEmpty()) {

      //make it the last link
      last = link;
   } else {

      //update first prev link
```

```
      head->prev = link;
   }

   //point it to old first link
   link->next = head;

   //point first to new first link
   head = link;
}
void main(){
   insertFirst(1,10);
   insertFirst(2,20);
   insertFirst(3,30);
   insertFirst(4,1);
   insertFirst(5,40);
   insertFirst(6,56);
   printf("\nDoubly Linked List: ");
   printList();
}
```

Output :

Doubly Linked List: (6,56) (5,40) (4,1) (3,30) (2,20) (1,10)

Program 40 ::- WAP for singly circular linked list.

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int data;
    struct node *next;
};
struct node *head;

void beginsert ();
void lastinsert ();
void randominsert();
void begin_delete();
void last_delete();
void random_delete();
void display();
void search();
void main ()
{
    int choice =0;
    while(choice != 7)
```

```
    {
        printf("\n*********Main Menu*********\n");
        printf("\nChoose one option from the following list ...\n");
        printf("\n===============================================\n");
        printf("\n1.Insert in begining\n2.Insert at last\n3.Delete from
Beginning\n4.Delete from last\n5.Search for an element\n6.Show\n7.Exit\n");
        printf("\nEnter your choice?\n");
        scanf("\n%d",&choice);
        switch(choice)
        {
            case 1:
            beginsert();
            break;
            case 2:
            lastinsert();
            break;
            case 3:
            begin_delete();
            break;
            case 4:
            last_delete();
            break;
            case 5:
            search();
            break;
            case 6:
            display();
            break;
            case 7:
            exit(0);
            break;
            default:
            printf("Please enter valid choice..");
        }
    }
}
void beginsert()
{
    struct node *ptr,*temp;
    int item;
    ptr = (struct node *)malloc(sizeof(struct node));
    if(ptr == NULL)
    {
        printf("\nOVERFLOW");
    }
    else
    {
        printf("\nEnter the node data?");
```

```
        scanf("%d",&item);
        ptr -> data = item;
        if(head == NULL)
        {
            head = ptr;
            ptr -> next = head;
        }
        else
        {
            temp = head;
            while(temp->next != head)
                temp = temp->next;
            ptr->next = head;
            temp -> next = ptr;
            head = ptr;
        }
        printf("\nnode inserted\n");
    }

}
void lastinsert()
{
    struct node *ptr,*temp;
    int item;
    ptr = (struct node *)malloc(sizeof(struct node));
    if(ptr == NULL)
    {
        printf("\nOVERFLOW\n");
    }
    else
    {
        printf("\nEnter Data?");
        scanf("%d",&item);
        ptr->data = item;
        if(head == NULL)
        {
            head = ptr;
            ptr -> next = head;
        }
        else
        {
            temp = head;
            while(temp -> next != head)
            {
                temp = temp -> next;
            }
            temp -> next = ptr;
            ptr -> next = head;
```

```
        }

        printf("\nnode inserted\n");
    }

}

void begin_delete()
{
    struct node *ptr;
    if(head == NULL)
    {
        printf("\nUNDERFLOW");
    }
    else if(head->next == head)
    {
        head = NULL;
        free(head);
        printf("\nnode deleted\n");
    }

    else
    {   ptr = head;
        while(ptr -> next != head)
            ptr = ptr -> next;
        ptr->next = head->next;
        free(head);
        head = ptr->next;
        printf("\nnode deleted\n");

    }
}
void last_delete()
{
    struct node *ptr, *preptr;
    if(head==NULL)
    {
        printf("\nUNDERFLOW");
    }
    else if (head ->next == head)
    {
        head = NULL;
        free(head);
        printf("\nnode deleted\n");

    }
    else
    {
```

```
        ptr = head;
        while(ptr ->next != head)
        {
            preptr=ptr;
            ptr = ptr->next;
        }
        preptr->next = ptr -> next;
        free(ptr);
        printf("\nnode deleted\n");

    }
}

void search()
{
    struct node *ptr;
    int item,i=0,flag=1;
    ptr = head;
    if(ptr == NULL)
    {
        printf("\nEmpty List\n");
    }
    else
    {
        printf("\nEnter item which you want to search?\n");
        scanf("%d",&item);
        if(head ->data == item)
        {
        printf("item found at location %d",i+1);
        flag=0;
        }
        else
        {
        while (ptr->next != head)
        {
            if(ptr->data == item)
            {
                printf("item found at location %d ",i+1);
                flag=0;
                break;
            }
            else
            {
                flag=1;
            }
            i++;
            ptr = ptr -> next;
        }
```

```
        }
        if(flag != 0)
        {
            printf("Item not found\n");
        }
    }

}

void display()
{
    struct node *ptr;
    ptr=head;
    if(head == NULL)
    {
        printf("\nnothing to print");
    }
    else
    {
        printf("\n printing values ... \n");

        while(ptr -> next != head)
        {

            printf("%d\n", ptr -> data);
            ptr = ptr -> next;
        }
        printf("%d\n", ptr -> data);
    }

}
```

Output :

```
      *********Main Menu*********

Choose one option from the following list ...

===============================================

1.Insert in begining
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Search for an element
6.Show
7.Exit
```

```
Enter your choice?
1

Enter the node data?10

node inserted

*********Main Menu*********

Choose one option from the following list ...

===============================================

1.Insert in begining
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Search for an element
6.Show
7.Exit

Enter your choice?
6

 printing values ...
10

*********Main Menu*********

Choose one option from the following list ...

===============================================

1.Insert in begining
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Search for an element
6.Show
7.Exit

Enter your choice?
2

Enter Data?20

node inserted
```

```
*********Main Menu*********

Choose one option from the following list ...

===============================================

1.Insert in begining
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Search for an element
6.Show
7.Exit

Enter your choice?
6

 printing values ...
10
20

*********Main Menu*********

Choose one option from the following list ...

===============================================

1.Insert in begining
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Search for an element
6.Show
7.Exit

Enter your choice?
1

Enter the node data?10

node inserted

*********Main Menu*********

Choose one option from the following list ...

===============================================
```

```
1.Insert in begining
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Search for an element
6.Show
7.Exit

Enter your choice?
1

Enter the node data?50

node inserted

*********Main Menu*********

Choose one option from the following list ...

================================================

1.Insert in begining
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Search for an element
6.Show
7.Exit

Enter your choice?
2

Enter Data?60

node inserted

*********Main Menu*********

Choose one option from the following list ...

================================================

1.Insert in begining
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Search for an element
6.Show
```

```
7.Exit

Enter your choice?
6

 printing values ...
50
10
10
20
60

*********Main Menu*********

Choose one option from the following list ...

================================================

1.Insert in begining
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Search for an element
6.Show
7.Exit

Enter your choice?
5

Enter item which you want to search?
50
item found at location 1
*********Main Menu*********

Choose one option from the following list ...

================================================

1.Insert in begining
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Search for an element
6.Show
7.Exit

Enter your choice?
7
```

Program 41 ::- WAP for doubly circular linked list code here

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
    struct node *prev;
    struct node *next;
    int data;
};
struct node *head;
void insertion_beginning();
void insertion_last();
void deletion_beginning();
void deletion_last();
void display();
void search();
void main ()
{
int choice =0;
    while(choice != 9)
    {
        printf("\n*********Main Menu*********\n");
        printf("\nChoose one option from the following list ...\n");
        printf("\n===============================================\n");
        printf("\n1.Insert in Beginning\n2.Insert at last\n3.Delete from
Beginning\n4.Delete from last\n5.Search\n6.Show\n7.Exit\n");
        printf("\nEnter your choice?\n");
        scanf("\n%d",&choice);
        switch(choice)
        {
            case 1:
            insertion_beginning();
            break;
            case 2:
                    insertion_last();
            break;
            case 3:
            deletion_beginning();
            break;
            case 4:
            deletion_last();
            break;
            case 5:
            search();
            break;
            case 6:
```

```
            display();
            break;
            case 7:
            exit(0);
            break;
            default:
            printf("Please enter valid choice..");
        }
    }
}
void insertion_beginning()
{
   struct node *ptr,*temp;
   int item;
   ptr = (struct node *)malloc(sizeof(struct node));
   if(ptr == NULL)
   {
       printf("\nOVERFLOW");
   }
   else
   {
    printf("\nEnter Item value");
    scanf("%d",&item);
    ptr->data=item;
   if(head==NULL)
   {
      head = ptr;
      ptr -> next = head;
      ptr -> prev = head;
   }
   else
   {
       temp = head;
    while(temp -> next != head)
    {
        temp = temp -> next;
    }
    temp -> next = ptr;
    ptr -> prev = temp;
    head -> prev = ptr;
    ptr -> next = head;
    head = ptr;
   }
   printf("\nNode inserted\n");
}

}
void insertion_last()
```

```
{
   struct node *ptr,*temp;
   int item;
   ptr = (struct node *) malloc(sizeof(struct node));
   if(ptr == NULL)
   {
       printf("\nOVERFLOW");
   }
   else
   {
       printf("\nEnter value");
       scanf("%d",&item);
        ptr->data=item;
       if(head == NULL)
       {
           head = ptr;
           ptr -> next = head;
           ptr -> prev = head;
       }
       else
       {
          temp = head;
          while(temp->next !=head)
          {
              temp = temp->next;
          }
          temp->next = ptr;
          ptr ->prev=temp;
          head -> prev = ptr;
      ptr -> next = head;
        }
   }
     printf("\nnode inserted\n");
}

void deletion_beginning()
{
    struct node *temp;
    if(head == NULL)
    {
        printf("\n UNDERFLOW");
    }
    else if(head->next == head)
    {
        head = NULL;
        free(head);
        printf("\nnode deleted\n");
    }
```

```
    else
    {
        temp = head;
        while(temp -> next != head)
        {
            temp = temp -> next;
        }
        temp -> next = head -> next;
        head -> next -> prev = temp;
        free(head);
        head = temp -> next;
    }

}
void deletion_last()
{
    struct node *ptr;
    if(head == NULL)
    {
        printf("\n UNDERFLOW");
    }
    else if(head->next == head)
    {
        head = NULL;
        free(head);
        printf("\nnode deleted\n");
    }
    else
    {
        ptr = head;
        if(ptr->next != head)
        {
            ptr = ptr -> next;
        }
        ptr -> prev -> next = head;
        head -> prev = ptr -> prev;
        free(ptr);
        printf("\nnode deleted\n");
    }
}

void display()
{
    struct node *ptr;
    ptr=head;
    if(head == NULL)
    {
        printf("\nnothing to print");
```

```
    }
    else
    {
        printf("\n printing values ... \n");

        while(ptr -> next != head)
        {

            printf("%d\n", ptr -> data);
            ptr = ptr -> next;
        }
        printf("%d\n", ptr -> data);
    }

}

void search()
{
    struct node *ptr;
    int item,i=0,flag=1;
    ptr = head;
    if(ptr == NULL)
    {
        printf("\nEmpty List\n");
    }
    else
    {
        printf("\nEnter item which you want to search?\n");
        scanf("%d",&item);
        if(head ->data == item)
        {
        printf("item found at location %d",i+1);
        flag=0;
        }
        else
        {
        while (ptr->next != head)
        {
            if(ptr->data == item)
            {
                printf("item found at location %d ",i+1);
                flag=0;
                break;
            }
            else
            {
                flag=1;
            }
```

```
            i++;
            ptr = ptr -> next;
        }
        }
        if(flag != 0)
        {
            printf("Item not found\n");
        }
    }

}
```

Output :

```
*********Main Menu*********

Choose one option from the following list ...

================================================

1.Insert in Beginning
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Search
6.Show
7.Exit

Enter your choice?
1

Enter Item value10

Node inserted

*********Main Menu*********

Choose one option from the following list ...

================================================

1.Insert in Beginning
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Search
6.Show
7.Exit
```

```
Enter your choice?
1

Enter Item value20

Node inserted

*********Main Menu*********

Choose one option from the following list ...

================================================

1.Insert in Beginning
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Search
6.Show
7.Exit

Enter your choice?
2

Enter value30

node inserted

*********Main Menu*********

Choose one option from the following list ...

================================================

1.Insert in Beginning
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Search
6.Show
7.Exit

Enter your choice?
2

Enter value40
```

```
node inserted

*********Main Menu*********

Choose one option from the following list ...

===============================================

1.Insert in Beginning
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Search
6.Show
7.Exit

Enter your choice?
6

 printing values ...
20
10
30
40

*********Main Menu*********

Choose one option from the following list ...

===============================================

1.Insert in Beginning
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Search
6.Show
7.Exit

Enter your choice?
3

*********Main Menu*********

Choose one option from the following list ...

===============================================
```

```
1.Insert in Beginning
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Search
6.Show
7.Exit

Enter your choice?
20
Please enter valid choice..
*********Main Menu*********

Choose one option from the following list ...

==============================================

1.Insert in Beginning
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Search
6.Show
7.Exit

Enter your choice?
6

 printing values ...
10
30
40

*********Main Menu*********

Choose one option from the following list ...

==============================================

1.Insert in Beginning
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Search
6.Show
7.Exit

Enter your choice?
```

```
1

Enter Item value100

Node inserted

*********Main Menu*********

Choose one option from the following list ...

================================================

1.Insert in Beginning
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Search
6.Show
7.Exit

Enter your choice?
6

 printing values ...
100
10
30
40

*********Main Menu*********

Choose one option from the following list ...

================================================

1.Insert in Beginning
2.Insert at last
3.Delete from Beginning
4.Delete from last
5.Search
6.Show
7.Exit

Enter your choice?
7
```

Day 7

Program 42 ::-

```c
#include<stdio.h>
#include<conio.h>

void main()
{
      name[30];
    printf("\n Enter the name : ");
    fgets(name,sizeof(name),stdin);

    printf("\n Your name is : %s ",name);
    printf("\n with puts function : ");
    puts(name);

    getch();
}
```

```
Output :
 Enter the name : z

 Your name is : z

 with puts function : z
```

Program 43 ::- Doubly linked list in c

```c
#include <stdio.h>
#include <stdlib.h>

struct node {
    char info;
    struct node *Ist;
    struct node *rst;
};

int main() {

    struct node *head = NULL;
    struct node *one = NULL;
    struct node *two = NULL;
    struct node *three = NULL;
```

```
    one = (struct node *)malloc(sizeof(struct node));
    two = (struct node *)malloc(sizeof(struct node));
    three = (struct node *)malloc(sizeof(struct node));

    one->info = 'A';
    two->info = 'B';
    three->info = 'C';

    one->Ist = NULL;
    one->rst = two;
    two->Ist = one;
    two->rst = three;
    three->Ist = two;
    three->rst = NULL;

    head = one;

    printf("Doubly linked list: ");
    struct node *current = head;
    while (current != NULL) {
        printf("%c ", current->info);
        current = current->rst;
    }
    printf("\n");

    free(one);
    free(two);
    free(three);

    return 0;
}
```

```
Output :
Doubly linked list: A B C
```

Program 44 ::- Doubly linked list with address

```
#include <stdio.h>
#include <stdlib.h>

struct Node {
    int data;
    struct Node *left;
    struct Node *right;
};

struct Node *head;
```

```
void create(int data){
    struct Node * nn=(struct Node *)malloc(sizeof(struct Node));
    struct Node *h=head;
    nn->data=data;
    nn->left=NULL;
    nn->right=NULL;
    if(h==NULL){
        head=nn;
        return ;
    }
    while(h->right!=NULL){
        h=h->right;
    }
    h->right=nn;
    nn->left=h;
}
void display(){
    struct Node *h=head;
    while(h!=NULL){
        printf("Left Address is :: %u || Value ::%d || Right Address is ::%u
|| Current Address :: %u\n",h->left,h->data,h->right,h);
        h=h->right;
    }
}
int main(){
    create(10);
    create(23);
    create(30);
    create(40);
    printf("Printing the data...\n");
    display();
}
```

Output :

Printing the data...
Left Address is :: 0 || Value ::10 || Right Address is ::12855768 || Current Address :: 12857296
Left Address is :: 12857296 || Value ::23 || Right Address is ::12855792 || Current Address :: 12855768
Left Address is :: 12855768 || Value ::30 || Right Address is ::12855816 || Current Address :: 12855792
Left Address is :: 12855792 || Value ::40 || Right Address is ::0 || Current Address :: 12855816

Program 45 ::-

```
// Binary tree traversal in C

#include <stdio.h>
```

```c
#include <stdlib.h>

struct node {
  int item;
  struct node* left;
  struct node* right;
};

// Inorder traversal
void inorderTraversal(struct node* root) {
  if (root == NULL) return;
  inorderTraversal(root->left);
  printf("%d ->", root->item);
  inorderTraversal(root->right);
}

// Preorder traversal
void preorderTraversal(struct node* root) {
  if (root == NULL) return;
  printf("%d ->", root->item);
  preorderTraversal(root->left);
  preorderTraversal(root->right);
}

// Postorder traversal
void postorderTraversal(struct node* root) {
  if (root == NULL) return;
  postorderTraversal(root->left);
  postorderTraversal(root->right);
  printf("%d ->", root->item);
}

// Create a new Node
struct node* createNode(value) {
  struct node* newNode = malloc(sizeof(struct node));
  newNode->item = value;
  newNode->left = NULL;
  newNode->right = NULL;

  return newNode;
}

// Insert on the left of the node
struct node* insertLeft(struct node* root, int value) {
  root->left = createNode(value);
  return root->left;
}
```

```c
// Insert on the right of the node
struct node* insertRight(struct node* root, int value) {
  root->right = createNode(value);
  return root->right;
}

int main() {
  struct node* root = createNode(1);
  insertLeft(root, 2);
  insertRight(root, 3);
  insertLeft(root->left, 4);

  printf("Inorder traversal \n");
  inorderTraversal(root);

  printf("\nPreorder traversal \n");
  preorderTraversal(root);

  printf("\nPostorder traversal \n");
  postorderTraversal(root);
}
```

```
Output :
Inorder traversal
4 ->2 ->1 ->3 ->
Preorder traversal
1 ->2 ->4 ->3 ->
Postorder traversal
4 ->2 ->3 ->1 ->
```

Day 8

Program 46 ::- sizeof array calculated using pointer

```c
#include<stdio.h>

void main()
{
    int arr[] = {1,2,3,4,5};

    int *ptr_arr[] = {arr, arr+1,arr+2,arr+3,arr+4};

    size_t size_of_ptr_arr = sizeof(ptr_arr);

    printf("Size of pointer array : %zu\n",size_of_ptr_arr);
}
```

Output :

```
Size of pointer array : 20
```

Program 47 ::- sizeof array and each element size is calculated.

```c
#include<stdio.h>

void main()
{
    int arr1[5];
    char arr2[5];
    double arr3[5];

    int *ptr1;
    ptr1 = arr1;

    char *ptr2;
    ptr2 = arr2;

    double *ptr3;
    ptr3 = arr3;

    printf("sizeof of arr1 of int is = %d \n",sizeof(arr1));
    printf("sizeof ptr1 of int is = %d \n",sizeof(ptr1));

    printf("sizeof of arr2 of char is = %d \n",sizeof(arr2));
    printf("sizeof ptr2 of char is = %d \n",sizeof(ptr1));

    printf("sizeof of arr3 of double is = %d \n",sizeof(arr3));
```

```
    printf("sizeof ptr3 of double is = %d \n",sizeof(ptr3));

}
```

```
Output :

sizeof of arr1 of int is = 20
sizeof ptr1 of int is = 4
sizeof of arr2 of char is = 5
sizeof ptr2 of char is = 4
sizeof of arr3 of double is = 40
sizeof ptr3 of double is = 4
```

Program 48 ::- size of operator in pointer

```
#include <stdio.h>

int main() {
    int array[5];
    int *ptr;

    ptr = array;
    printf("Size of the array                          : %zu bytes\n",
sizeof(array));
    printf("Size of the pointer                        : %zu bytes\n",
sizeof(ptr));

    printf("Size of an element in the array            : %zu bytes\n",
sizeof(array[0]));
    printf("Size of an element pointed to by the pointer: %zu bytes\n",
sizeof(ptr[0]));


    printf("Number of elements in the array            : %zu\n",
sizeof(array) / sizeof(array[0]));

    return 0;
}
```

```
Output :

Size of the array                : 20 bytes
Size of the pointer               : 4 bytes
```

```
Size of an element in the array          : 4 bytes
Size of an element pointed to by the pointer: 4 bytes
Number of elements in the array          : 5
```

Program 49 ::- sizeof structure calculated

```c
#include <stdio.h>

struct name {
    int member1;
    int member2;

};

int main()
{
    struct name *ptr, Harry;

    printf("Size of struct name: %zu bytes\n",sizeof(struct name));
    printf("Size of member1: %zu bytes\n", sizeof(Harry.member1));
    printf("Size of member2: %zu bytes\n", sizeof(Harry.member2));
    ptr=&Harry;
    printf("addres of name %zu\n",ptr);
    ptr=&Harry.member1;
    printf("address of Harry %zu\n",ptr);

     ptr=&Harry.member2;
    printf("address of Harry %zu\n",ptr);


    return 0;
}
```

Output :

```
Size of struct name: 8 bytes
Size of member1: 4 bytes
Size of member2: 4 bytes
addres of name 6422292
address of Harry 6422292
address of Harry 6422296
```

Program 50 ::- Concept of structure in c

```c
#include <stdio.h>
struct person
{   int age;
   float weight;
};

int main()
{
    struct person *personPtr, person1;
    personPtr = &person1;
    printf("Enter age: ");
    scanf("%d", &personPtr->age);

    printf("Enter weight: ");
    scanf("%f", &personPtr->weight);

    printf("Displaying:\n");
    printf("Age: %d\n", personPtr->age);
    printf("weight: %f", personPtr->weight);

    return 0;
}
```

Output :

```
Enter age: 23
Enter weight: 50
Displaying:
Age: 23
weight: 50.000000
```

Program 51 ::- Concept of structure in c using dynamic memory allocation

```c
#include <stdio.h>
#include <stdlib.h>

struct person {
   int age;
   float weight;
   char name[30];
};

int main()
{
   struct person *ptr;
```

```c
    int i, n;

    printf("Enter the number of persons: ");
    scanf("%d", &n);

    // allocating memory for n numbers of struct person
    ptr = (struct person*) malloc(n * sizeof(struct person));

    for(i = 0; i < n; ++i)
    {
        printf("Enter first name and age respectively: ");

        scanf("%s %d", (ptr+i)->name, &(ptr+i)->age);
    }

    printf("Displaying Information:\n");
    for(i = 0; i < n; ++i)
        printf("Name: %s\tAge: %d\n", (ptr+i)->name, (ptr+i)->age);

    return 0;
}
```

```
Output :

Enter the number of persons: 4
Enter first name and age respectively: o
2
Enter first name and age respectively: p
3
Enter first name and age respectively: q
4
Enter first name and age respectively: r
5
Displaying Information:
Name: o Age: 2
Name: p Age: 3
Name: q Age: 4
Name: r Age: 5
```

Program 52 ::- Concept of structure in c using dynamic memory allocation

```c
#include <stdio.h>
struct student {
    char name[50];
    int age;
};
```

```c
// function prototype
void display(struct student s);

int main() {
   struct student s1;

   printf("Enter name: ");

   // read string input from the user until \n is entered
   // \n is discarded
   scanf("%[^\n]%*c", s1.name);

   printf("Enter age: ");
   scanf("%d", &s1.age);

   display(s1); // passing struct as an argument

   return 0;
}

void display(struct student s) {
   printf("\nDisplaying information\n");
   printf("Name: %s", s.name);
   printf("\nAge: %d", s.age);
}
```

Output :

Enter name: Vaibhav
Enter age: 24

Displaying information
Name: Vaibhav
Age: 24

Program 53 ::- Concept of union in c.

```c
#include <stdio.h>

union Job
{
   float salary;
   int workerNo;
   char name[20];
```

```
} j;

int main()
{
   j.salary = 12.3;

   // when j.workerNo is assigned a value,// j.salary will no longer hold
12.3   j.workerNo = 100;

   printf("Salary = %.1f\n", j.salary);
   printf("Number of workers = %d\n", j.workerNo);
   printf("sizeof union %zu\n", sizeof(j));

   return 0;
}
```

```
Output :

Salary = 12.3
Number of workers = 1095027917
sizeof union 20
```

Day 9

Program 54 ::- WAP to check number is prime or not

```c
#include<stdio.h>
    int main()
{
    int n,i,m=0,flag=0;

    printf("Enter the number to check prime:");
    scanf("%d",&n);

    m=n/2;
    for(i=2;i<=m;i++)
    {
        if(n%i==0)
        {
            printf("Number is not prime");
            flag=1;
            break;
        }
    }
    if(flag==0)
    printf("Number is prime");
    return 0;
    }
```

Output : Enter the number to check prime:13
Number is prime

Program 55 ::- WAP to check number is prime or not

```c
#include<stdio.h>
int main(){
    int a;
    scanf("%d",&a);
    prime(a);
}
void prime(int a){
    if(a<=1){
     printf("%d is not prime",a);
     return ;
    }
    int c=0;
    for(int i=1;i<=a;i++){
        if(a%i==0){
```

```
            c++;
        }
    }
    if(c==2){
        printf("%d is prime",a);
    }
    else
        printf("%d is not  prime",a);

}
```

```
Output :
12
12 is not  prime
```

Program 56 ::- WAP for factorial of number using recursion.

```
#include <stdio.h>
int sum(int n);

int main()
{
    int number, result;

    printf("Enter a positive integer: ");
    scanf("%d", &number);

    result = sum(number);
    printf("sum = %d", result);

    return 0;
}

int sum(int n) {
    if (n != 0)
        return n + sum(n-1);
        // sum() function calls itselfreturn n + sum(n-1);
    else
        return n;
}
```

```
Output : Enter a positive integer: 5
sum = 15
```

Program 57 ::- Tower of Hanoi by using recursion.

```c
#include <stdio.h>
void hanoi(int n, char from, char to, char via) {
   if(n == 1){
      printf("Move disk 1 from %c to %c\n", from, to);
   }
   else{
      hanoi(n-1, from, via, to);
      printf("Move disk %d from %c to %c\n", n, from, to);
      hanoi(n-1, via, to, from);
   }
}
int main() {
   int n = 3;
   char from = 'A';
   char to = 'B';
   char via = 'C';
   //calling hanoi() method
   hanoi(n, from, via, to);
}
```

```
Output :
Move disk 1 from A to C
Move disk 2 from A to B
Move disk 1 from C to B
Move disk 3 from A to C
Move disk 1 from B to A
Move disk 2 from B to C
Move disk 1 from A to C
```

Day 10

Program 58 ::- AVL tree implementation in C

```c
#include <stdio.h>
#include <stdlib.h>

// Create Node
struct Node {
  int key;
  struct Node *left;
  struct Node *right;
  int height;
};

int max(int a, int b);

// Calculate height
int height(struct Node *N) {
  if (N == NULL)
    return 0;
  return N->height;
}

int max(int a, int b) {
  return (a > b) ? a : b;
}

// Create a node
struct Node *newNode(int key) {
  struct Node *node = (struct Node *)
    malloc(sizeof(struct Node));
  node->key = key;
  node->left = NULL;
  node->right = NULL;
  node->height = 1;
  return (node);
}

// Right rotate
struct Node *rightRotate(struct Node *y) {
  struct Node *x = y->left;
  struct Node *T2 = x->right;

  x->right = y;
  y->left = T2;
```

```
  y->height = max(height(y->left), height(y->right)) + 1;
  x->height = max(height(x->left), height(x->right)) + 1;

  return x;
}

// Left rotate
struct Node *leftRotate(struct Node *x) {
  struct Node *y = x->right;
  struct Node *T2 = y->left;

  y->left = x;
  x->right = T2;

  x->height = max(height(x->left), height(x->right)) + 1;
  y->height = max(height(y->left), height(y->right)) + 1;

  return y;
}

// Get the balance factor
int getBalance(struct Node *N) {
  if (N == NULL)
    return 0;
  return height(N->left) - height(N->right);
}

// Insert node
struct Node *insertNode(struct Node *node, int key) {
  // Find the correct position to insertNode the node and insertNode it
  if (node == NULL)
    return (newNode(key));

  if (key < node->key)
    node->left = insertNode(node->left, key);
  else if (key > node->key)
    node->right = insertNode(node->right, key);
  else
    return node;

  // Update the balance factor of each node and
  // Balance the tree
  node->height = 1 + max(height(node->left),
                         height(node->right));

  int balance = getBalance(node);
  if (balance > 1 && key < node->left->key)
    return rightRotate(node);
```

```
  if (balance < -1 && key > node->right->key)
    return leftRotate(node);

  if (balance > 1 && key > node->left->key) {
    node->left = leftRotate(node->left);
    return rightRotate(node);
  }

  if (balance < -1 && key < node->right->key) {
    node->right = rightRotate(node->right);
    return leftRotate(node);
  }

  return node;
}

struct Node *minValueNode(struct Node *node) {
  struct Node *current = node;

  while (current->left != NULL)
    current = current->left;

  return current;
}

// Delete a nodes
struct Node *deleteNode(struct Node *root, int key) {
  // Find the node and delete it
  if (root == NULL)
    return root;

  if (key < root->key)
    root->left = deleteNode(root->left, key);

  else if (key > root->key)
    root->right = deleteNode(root->right, key);

  else {
    if ((root->left == NULL) || (root->right == NULL)) {
      struct Node *temp = root->left ? root->left : root->right;

      if (temp == NULL) {
        temp = root;
        root = NULL;
      } else
        *root = *temp;
      free(temp);
```

```
    } else {
      struct Node *temp = minValueNode(root->right);

      root->key = temp->key;

      root->right = deleteNode(root->right, temp->key);
    }
  }

  if (root == NULL)
    return root;

  // Update the balance factor of each node and
  // balance the tree
  root->height = 1 + max(height(root->left),
                         height(root->right));

  int balance = getBalance(root);
  if (balance > 1 && getBalance(root->left) >= 0)
    return rightRotate(root);

  if (balance > 1 && getBalance(root->left) < 0) {
    root->left = leftRotate(root->left);
    return rightRotate(root);
  }

  if (balance < -1 && getBalance(root->right) <= 0)
    return leftRotate(root);

  if (balance < -1 && getBalance(root->right) > 0) {
    root->right = rightRotate(root->right);
    return leftRotate(root);
  }

  return root;
}

// Print the tree
void printPreOrder(struct Node *root) {
  if (root != NULL) {
    printf("%d ", root->key);
    printPreOrder(root->left);
    printPreOrder(root->right);
  }
}

int main() {
  struct Node *root = NULL;
```

```
  root = insertNode(root, 2);
  root = insertNode(root, 1);
  root = insertNode(root, 7);
  root = insertNode(root, 4);
  root = insertNode(root, 5);
  root = insertNode(root, 3);
  root = insertNode(root, 8);

  printPreOrder(root);

  root = deleteNode(root, 3);

  printf("\nAfter deletion: ");
  printPreOrder(root);

  return 0;
}
```

```
Output :
4 2 1 3 7 5 8
After deletion: 4 2 1 7 5 8
```