

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
GRAPHIC ERA HILL UNIVERSITY, BHIMTAL CAMPUS
SATTAL ROAD, P.O. BHOWALI



DATA STRUCTURE USING C
(PCS 302)
(2022-2023)

Lab File
For
3rd Semester

Submitted To:

Mr. Devesh Pandey
(Assistant Professor)
Department OF CSE

Submitted By:

Ms.Hemant Kumar Gaur
B. Tech CSE (3rd Sem)
Roll Number:14

1. Write a program to store n elements in an array, and then check how many elements are palindrome numbers in that array.

```
#include<stdio.h>

int main()
{
    int a[10],i,j,s,n,t;
    printf("\nEnter the element :\n");
    for(i=0;i<10;i++)
    {
        scanf("%d",&a[i]);
    }
    for(i=0;i<10;i++)
    {
        n=a[i];
        s=0;
        while(n>0)
        {
            t=n%10;
            s=s*10+t;
            n=n/10;
        }
        if(s==a[i])
        {
            printf("\n %d is palindrome",a[i]);
        }
    }
    return 0;}
```

454 is palindrome
0 is palindrome

2. Write a program to enter n numbers and print them using malloc()
[Example of dynamic array].

```
#include <stdio.h>

#include <stdlib.h>

int main()

{

    int n, i, *ptr, sum = 0;


    printf("Enter number of elements: ");

    scanf("%d", &n);


    ptr = (int*) malloc(n * sizeof(int));


    if(ptr == NULL)

    {

        printf("Error! memory not allocated.");

        exit(0);

    }


    printf("Enter elements: ");

    for(i = 0; i < n; ++i)

    {

        scanf("%d", ptr + i);

        sum += *(ptr + i);

    }

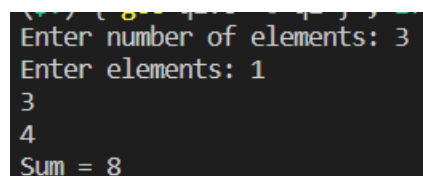

    printf("Sum = %d", sum);

    free(ptr);

    return 0;

}
```

OUTPUT:-

A screenshot of a terminal window showing the output of the program. The text is as follows:

```
Enter number of elements: 3
Enter elements: 1
3
4
Sum = 8
```

3. Write a program to dynamically allocate the records of n employees using calloc(). Insert the records and print them. T

```
#include<stdio.h>
#include<stdlib.h>
typedef struct emp{
    char name[30];
    int age;
    double salary;
}data;

int main(){
    int n,i;
    printf("ENTER THE NUMBER OF EMPLOYEES :- \n");
    scanf("%d",&n);
    data *emp_data;
    emp_data=(data*)calloc(sizeof(data),n);
    for(i=0;i<n;i++){
        printf("ENTER THE NAME \n");
        fflush(stdin);
        gets(emp_data[i].name);

        printf("ENTER THE SALARY \n");
        scanf("%d",&emp_data[i].salary);
        printf("ENTER THE AGE\n");
        scanf("%d",&emp_data[i].age);

    }
    for(i=0;i<n;i++){
        printf("\nTHE DETAILS ARE \n%s\n%d\n%d",emp_data[i].name,emp_data[i].salary,emp_data[i].age);
    }
}
```

OUTPUT:-

```
ENTER THE NUMBER OF EMPLOYEES :-
50
ENTER THE NAME
simran
ENTER THE SALARY
200000-
ENTER THE AGE
22
ENTER THE NAME
ENTER THE SALARY
```

4. Write a menu driven program to input a 2-D array, output it, add all its elements, add only the diagonal elements, count the number of odd elements and find the largest among the elements stored

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

```
void input(int arr[20][20], int n)
{
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            scanf("%d", &arr[i][j]);
        }
    }
}
```

```
void output(int arr[20][20], int n)
{
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            printf("%d ", arr[i][j]);
        }
        printf("\n");
    }
}
```

```
void Add(int arr[20][20], int n)
{
    int sum = 0;
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
```

```

        {
            sum += arr[i][j];
        }
    }
    printf("Sum of all elements: %d\n", sum);
}

```

```

void AddDiagnol(int arr[20][20], int n)
{
    int sumL = 0, sumR = 0;
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if (i == j)
                sumL += arr[i][j];
            if (i + j == n - 1)
                sumR += arr[i][j];
        }
    }
    printf("Sum of all diagnols: %d\n", sumL + sumR);
}

```

```

void countOdd(int arr[20][20], int n)
{
    int count = 0;
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if ((arr[i][j]) % 2 != 0)
                count++;
        }
    }
}

```

```

    }

    printf("No of odd elements\n", count);
}

void largestElement(int arr[20][20], int n)
{
    int maxi = arr[0][0];
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if (arr[i][j] >= maxi)
                maxi = arr[i][j];
        }
    }

    printf("Largest Element in array: %d \n", maxi);
}

int main()
{
    int n, arr[20][20], ch;

    printf("Enter the m X n range for matrix\n");
    scanf("%d", &n);

    while (1)
    {
        printf("Enter your choice:\n");
        printf("1.Input\n");
        printf("2.Output\n");
        printf("3.Add All Elements\n");
        printf("4.Add Diagnol Elements\n");
        printf("5.Count no of odd elements\n");
        printf("6.Largest Element\n");
        printf("7.Exit\n");
    }
}

```

```

scanf("%d", &ch);

switch (ch)
{
case 1:
    input(arr, n);
    break;
case 2:
    output(arr, n);
    break;
case 3:
    Add(arr, n);
    break;
case 4:
    AddDiagnol(arr, n);
    break;
case 5:
    countOdd(arr, n);
    break;
case 6:
    largestElement(arr, n);
    break;
case 7:
    return 0;
}
}
return 0;
}

```

```

Enter the m X n range for matrix
3 3
Enter your choice:
1.Input
2.Output
3.Add All Elements
4.Add Diagnol Elements
5.Count no of odd elements
6.Largest Element
7.Exit

```


5. Write a program to read integers into an array and reverse those using pointers.

```
• #include<stdio.h>
• #include<conio.h>
• #define MAX 30
• void main() {
•     int size,i,arr[MAX];
•     int *ptr;
•     ptr = &arr[0];
•     printf("\nEnter the size of array : ");
•     scanf("%d",&size);
•     printf("\nEnter %d integers into array: ",size);
•     for (i=0;i<size;i++) {
•         scanf("%d",ptr);
•         ptr++;
•     }
•     ptr=&arr[size-1];
•     printf("\nElements of array in reverse order are :");
•
•     for (i=size-1;i>=0;i--) {
•         printf("\nElement%d is %d : ",i,*ptr);
•         ptr--;
•     }
•     return 0;
• }
```

```
Enter the size of array : 4
Enter 4 integers into array: 1 2 3 4
Elements of array in reverse order are :
Element3 is 4 :
Element2 is 3 :
Element1 is 2 :
Element0 is 1 :
```

6. . Write a program to calculate sum of upper triangular elements.

```
• #include<stdio.h>
• #include<conio.h>
• int main() {
•     int i, j, a[10][10],sum,rows,columns;
•     printf("\nEnter the number of Rows : ");
•     scanf("%d",&rows);
•
•     printf("\nEnter the number of Columns : ");
•     scanf("%d",&columns);
•
•     for (i=0;i<rows;i++)
•         for (j=0; j<columns;j++)
•         {
•             printf("\nEnter the Element a[%d][%d] : ",i,j);
•             scanf("%d", &a[i][j]);
•         }
•     sum = 0;
•     for (i=0;i<rows;i++)
•         for (j=0;j<columns;j++)
•         {
•             if (i<j)
•             {
•                 sum=sum+a[i][j];
•             }
•         }
•     printf("\nSum of Upper Triangle Elements : %d",sum);
•     return 0;
• }
```

Enter the number of Rows : 4

Enter the number of Columns : 4

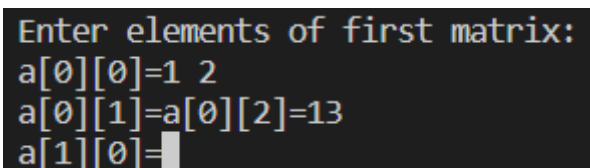
Enter the Element a[0][0] : 1 2 3 4 5 6 7 8

Enter the Element a[0][1] :

Enter the Element a[0][2] :

7. Write a program to multiply two 3x3 matrices

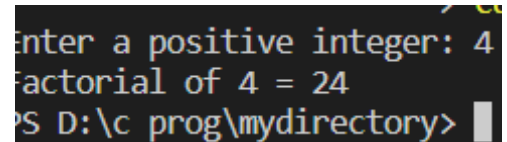
```
• #include<stdio.h>
• int main()
• {
•     int i,j,k;
•     float a[3][3],b[3][3],mul[3][3];
•
•     printf("Enter elements of first matrix:\n");
•     for(i=0;i<3;i++)
•     {
•         for(j=0;j<3;j++)
•         {
•             printf("a[%d][%d]=",i,j);
•             scanf("%f",&a[i][j]);
•         }
•     }
•     {
•         for(j=0;j<3;j++)
•         {
•             mul[i][j]=0;
•             for(k=0;k<3;k++)
•             {
•                 mul[i][j]=mul[i][j]+a[i][k]*b[j][k];
•             }
•         }
•     }
•     printf("Multiplied matrix is:\n");
•     for(i=0;i<3;i++){
•         for(j=0;j<3;j++){
•             printf("%f\t", mul[i][j])
•         }
•         printf("\n");
•     }
•     Return 0;
• }
```



```
Enter elements of first matrix:
a[0][0]=1 2
a[0][1]=a[0][2]=13
a[1][0]=
```

8. Write a program to find the factorial of any number using linear recursion.

```
• #include<stdio.h>
• long int multiplyNumbers(int n);
• int main()
• {
•     int n;
•     printf("Enter a positive integer: ");
•     scanf("%d",&n);
•     printf("Factorial of %d = %ld",n,multiplyNumbers(n));
•     return 0;
• }
•
long int multiplyNumbers(int n)
• {
•     if (n>=1)
•         return n*multiplyNumbers(n-1);
•     else
•         return 1;
• }
```



enter a positive integer: 4
Factorial of 4 = 24
PS D:\c prog\mydirectory>

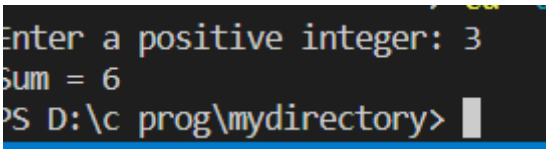
9. Write a program to find the factorial of any number using tail recursion.

```
• #include<stdio.h>
• int factorial(int n,int fact)
• {
•     if ( n==1 )
•         return fact;
•     else
•         factorial(n-1,n*fact);
• }
•
int main( ){
•     int n,value;
•     printf("Enter the number : ");
•     scanf( "%d",&n);
•
•     if ( n < 0 )
•         printf("No factorial of negative number\n");
•     else if ( n==0 )
•         printf("Factorial of zero is 1\n");
•     else
•     {
•         value = factorial(n,1);
•         printf("Factorial of %d = %d\n",n,value);
•     }
•     return 0;
• }
```

```
Enter the number : 4
Factorial of 4 = 24
PS D:\c prog\mydirectory> |
```

10. Program to find sum of n natural numbers using tail recursion.

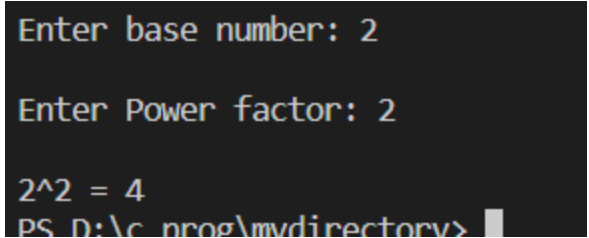
```
• #include <stdio.h>
• int addNumbers(int n);
• int main() {
•
    int num;
•     printf("Enter a positive integer: ");
•     scanf("%d", &num);
•     printf("Sum = %d", addNumbers(num));
•     return 0;
• }
•
int addNumbers(int n) {
•     if (n != 0)
•         return n+addNumbers(n - 1);
•     else
•         return n;
• }
```



```
Enter a positive integer: 3
Sum = 6
PS D:\c prog\mydirectory>
```

11. Write a program to find nm using linear recursion.

```
• #include<stdio.h>
• int power(int n1, int n2);
• int main()
• {
•     int base, exp;
•     printf("Enter base number: ");
•     scanf("%d",&base);
•     printf("\nEnter Power factor: ");
•     scanf("%d",&exp);
•     printf("\n%d^%d = %d", base,exp,power(base, exp));
•     return 0;
• }
•
int power(int b, int e)
• {
•     if(e == 0)
•         return 1;
•
•     return (b*power(b, e-1));
• }
```



```
Enter base number: 2
```

```
Enter Power factor: 2
```

```
2^2 = 4
```

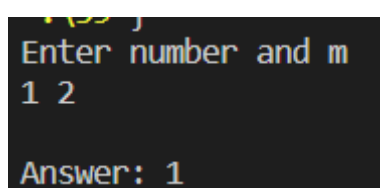
```
PS D:\c prog\mydirectory>
```

12. Write a program to find nm using tail recursion

```
#include<stdio.h>
#include<stdlib.h>
int ans=1;
void tail(int n, int m, int x)
{
    if(m==x)
        return;

    tail(n,m,x+1);
    ans = ans*n;
}
int main()
{
    int n,m;
    printf("Enter number and m\n");
    scanf("%d %d", &n, &m);

    tail(n, m, 0);
    printf("\nAnswer: %d\n", ans);
return 0;
}
```

A screenshot of a terminal window showing the execution of the program. The prompt is 12. The user enters '1 2' in response to the prompt 'Enter number and m'. The program then outputs 'Answer: 1'.

```
12. Enter number and m
1 2
Answer: 1
```

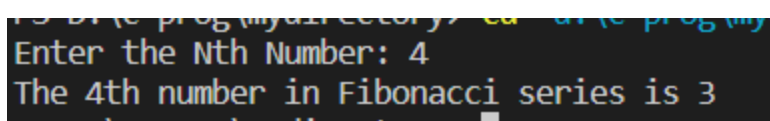

13. Write a program to find the maximum among array elements recursively

```
• #include <stdio.h>
• int find_maximum(int[],int);
• int main()
• {
•     int c, array[100], size,location,maximum;
    printf("Input number of elements in array\n");
•     scanf("%d",&size);
    printf("Enter %d integers\n",size);
    for (c = 0;c<size;c++)
•         scanf("%d",&array[c]);
    location=find_maximum(array,size);
•     maximum=array[location];
•
    printf("Maximum element location = %d and value =
%d.\n",location+1,maximum);
•     return 0;
• }
int find_maximum(int a[],int n) {
•     int c,max,index;
    max = a[0];
•     index = 0;
    for (c=1;c<n;c++) {
•         if (a[c]>max) {
•             index=c;
•             max=a[c];
•         }
•     }
•     return index;
• }
```

```
Input number of elements in array
4
Enter 4 integers
1 2 3 4
Maximum element location = 4 and value = 4.
```

14. Write a program to compute nthFibonacci number using binary recursion.

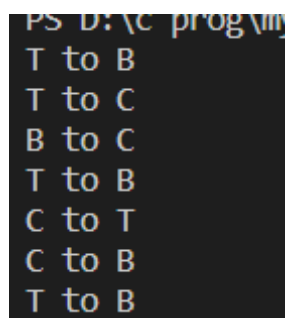
```
• #include<stdio.h>
• int fib (int);
• int main ()
• {
•     int n,res;
•     printf("Enter the Nth Number: ");
•     scanf("%d",&n);
•     res=fib(n);
•     printf ("The %dth number in Fibonacci series is %d\n",n,res);
•     return 0;
• }
•
int fib (int n)
• {
•     if (n==0)
•     {
•         return 0;
•     }
•     else if (n==1)
•     {
•         return 1;
•     }
•     else
•     {
•         return (fib(n-1)+fib(n-2));
•     }
• }
```



```
C:\> gcc prog14.c -o fib
Enter the Nth Number: 4
The 4th number in Fibonacci series is 3
```

15. Write a program of Tower of Hanoi to shift n plates from one peg to another using a temporary peg

```
• #include<stdio.h>
• void TOH(int n,char x,char y,char z) {
•     if(n>0) {
•         TOH(n-1,x,z,y);
•         printf("%c to %c",x,y);
•         TOH(n-1,z,y,x);
•     }
• }
• int main() {
•     int n=3;
•     TOH(n,'T','B','C');
• }
```



```
PS D:\c program>
T to B
T to C
B to C
T to B
C to T
C to B
T to B
```

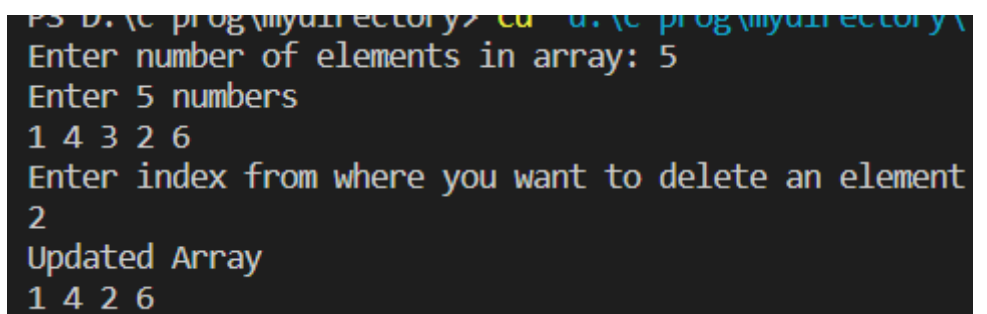
16. Write a program to insert an element at a given position in an array.

```
• #include <stdio.h>
• int main()
• {
•     int arr[100] = { 0 };
•     int i, x, pos, n = 10;
•
•     for (i = 0; i < 10; i++)
•     {
•         arr[i] = i + 1;
•     }
•     printf("Before Insertion :");
•     for (i = 0; i < n; i++)
•     {
•         printf("%d ", arr[i]);
•     }
•     printf("\n");
•     x = 50;
•     pos = 5;
•     n++;
•     for (i = n - 1; i >= pos; i--)
•     {
•         arr[i] = arr[i - 1];
•     }
•     arr[pos - 1] = x;
•     printf("After Insertion :");
•     for (i = 0; i < n; i++)
•     {
•         printf("%d ", arr[i]);
•     }
•     printf("\n");
•     return 0;
```

```
Before Insertion :1 2 3 4 5 6 7 8 9 10
After Insertion :1 2 3 4 50 5 6 7 8 9 10
```

17. Write a program to delete an element from an array.

```
• #include <stdio.h>
• #include <conio.h>
• int main(){
•     int a[500],n,i,index;
•     printf("Enter number of elements in array: ");
•     scanf("%d",&n);
•     printf("Enter %d numbers \n",n);
•     for(i=0;i<n;i++)
•
•         scanf("%d",&a[i]);
•     }
•     printf("Enter index from where you want to delete an element\n");
•     scanf("%d",&index);
•     for(i=index;i<n-1;i++){
•         a[i]=a[i+1];
•     }
•     printf("Updated Array\n");
•     for(i = 0;i<n-1;i++)
•     {
•         printf("%d ",a[i]);
•     }
•     return 0;
• }
```



```
PS D:\c prog\mydirectory> cd d:\c prog\mydirectory\
Enter number of elements in array: 5
Enter 5 numbers
1 4 3 2 6
Enter index from where you want to delete an element
2
Updated Array
1 4 2 6
```

18. Write a menu driven program having following functionalities: input an array, output, insert an element, delete an element, sort array in ascending order, sort an array in descending order [without taking global variable

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

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

```
void input(int arr[], int n)
```

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

```
void output(int arr[], int n)
```

```
{
    int i;
    for(i=0; i<n;i++)
        printf("%d ", arr[i]);
    printf("\n");
}
```

```
void sortAscending(int arr[], int n)
```

```
{
    int i,j;
    printf("Sort in ascending\n");
    for(i=0; i<n; i++)
    {
        for(j=i+1; j<n; j++)
        {
            if(arr[j] <= arr[i]){
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
    }
}
```

```

        }
    }
}

void insert(int arr[], int *n, int data)
{
    arr[*n] = data;
    *n=*n+1;
}

int delete(int arr[], int *n)
{
    int x = arr[*n-1];
    *n = *n -1;
    return x;
}

void sortDescending(int arr[], int n)
{
    int i,j;
    printf("Sort in descennding\n");
    for(i=0; i<n; i++)
    {
        for(j=i+1; j<n; j++)
        {
            if(arr[j] >= arr[i]){
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
    }
}

```

```

int main()
{
    int n;
    printf("Enter the number of elements\n");
    scanf("%d", &n);

    int arr[50];
    input(arr,n);
    output(arr, n);
    sortAscending(arr, n);
    output(arr, n);
    sortDescending(arr, n);
    output(arr, n);

    insert(arr, &n, 99);
    output(arr, n);
    delete(arr, &n);
    output(arr, n);
    return 0;
}

```

```

Enter the number of elements
4 3 2 1
5 6 7 8
3 2 1 5
Sort in ascending
1 2 3 5
Sort in descennding
5 3 2 1
5 3 2 1 99
5 3 2 1

```


19. Wap to first sort the elements stored in two arrays in ascending order then merge them into a third array [without taking global var

```
#include <Stdio.h>
#include <conio.h>

main() {
    int arr1 [10], arr2 [10], arr3 [20];
    int i, n1, n2, m, index=0;
    printf("\n Enter the number of elements in array 1: ");
    scanf("%d", &n1);
    printf("\nEnter the Elements of the first array ");
    for(i=0;i<n1;i++) {
        scanf ("%d",&arr1[i]);
    }
    printf("\nEnter the number of elements in array 2: ");
    scanf ("%d", &n2 );
    printf("\nEnter the Elements of the second array ");
    for(i=0;i<n2;i++) {
        scanf ("%d", &arr2[i]);
        m = n1+n2;
    }
    for(i=0;i<n1;i++) {
        arr3[index]=arr1[i];
        index++;
    }
    for(i=0;i<n2;i++) {
        arr3[index]=arr2[i];
        index++;
    }
    printf ("\nThe merged array is\n");
    for(i=0;i<m;i++) {
        printf("%d ",arr3[i]);
    }
    return 0; }
```

```
Enter the number of elements in array 1: 4
Enter the Elements of the first array 1 2 3 4
Enter the number of elements in array 2: 4 5 6 7
Enter the Elements of the second array 4 5 7 8
The merged array is
1 2 3 4 5 6 7 4
```

20. Write a menu driven program of a stack using array having the push, pop and display operations[without taking global variables

```
• #include<stdio.h>
• int stack[100],choice,n,top,x,i;
• void push();
• void pop();
• void display();
• int main()
• {
•     top=-1;
•     printf("Enter the size of STACK[MAX=100]:");
•     scanf("%d",&n);
•     printf("\nSTACK OPERATIONS USING ARRAY");
•     printf("\n-----");
•     printf("\n1.PUSH\n2.POP\n3.DISPLAY\n4.EXIT");
•     do
•     {
•         printf("\nEnter the Choice:");
•         scanf("%d",&choice);
•         switch(choice)
•         {
•             case 1:
•             {
•                 push();
•                 break;
•             }
•             case 2:
•             {
•                 pop();
•                 break;
•             }
•             case 3:
•             {
•                 display();
```

```

•             break;
•         }
•         case 4:
•         {
•             printf("\nEXIT POINT");
•             break;
•         }
•         default:
•         {
•             printf ("\nPlease Enter a Valid Choice(1/2/3/4)");
•         }
•
•     }
• }
• while(choice!=4);
• return 0;
• }
• void push()
• {
•     if(top>=n-1)
•     {
•         printf("\n\tSTACK is over flow");
•
•     }
•     else
•     {
•         printf(" Enter a value to be pushed:");
•         scanf("%d",&x);
•         top++;
•         stack[top]=x;
•     }
• }
• void pop()

```

```

• {
•     if(top<=-1)
•     {
•         printf("\n\t Stack is under flow");
•     }
•     else
•     {
•         printf("\n\t The popped elements is %d",stack[top]);
•         top--;
•     }
• }
• void display()
• {
•     if(top>=0)
•     {
•         printf("\n The elements in STACK \n");
•         for(i=top; i>=0; i--)
•             printf("\n%d",stack[i]);
•         printf("\n Press Next Choice");
•     }
•     else
•     {
•         printf("\n The STACK is empty");
•     }
• }

```

```

PS D:\c prog\mydirectory> cd d:\c prog\my
Enter the size of STACK[MAX=100]:7

STACK OPERATIONS USING ARRAY
-----
1.PUSH
2.POP
3.DISPLAY
4.EXIT
Enter the Choice:1
Enter a value to be pushed:1

Enter the Choice:1
Enter a value to be pushed:2

Enter the Choice:1
Enter a value to be pushed:3

Enter the Choice:1
Enter a value to be pushed:4

Enter the Choice:3

The elements in STACK

4
3
2
1
Press Next Choice
Enter the Choice:

```

21. Write a menu driven program of a queue using array having add, delete and display operations [without taking global variables

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

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

```
#define MAX 10
```

```
int queue_arr[MAX];
```

```
int rear=-1;
```

```
int front=-1;
```

```
void insert(int item);
```

```
int del();
```

```
int peek();
```

```
void display();
```

```
int isFull();
```

```
int isEmpty();
```

```
int main()
```

```
{
```

```
    int choice,item;
```

```
    while(1)
```

```
    {
```

```
        printf("1.Insert\n");
```

```
        printf("2.Delete\n");
```

```
        printf("3.Display element at the front\n");
```

```
        printf("4.Display all elements of the queue\n");
```

```
        printf("5.Quit\n");
```

```
        printf("\nEnter your choice : ");
```

```
        scanf("%d",&choice);
```

```
        switch(choice)
```

```
        {
```

```
            case 1:
```

```

        printf("\nInput the element for adding in queue
: ");

        scanf("%d",&item);
        insert(item);
        break;
    case 2:
        item=del();
        printf("\nDeleted element is  %d\n",item);
        break;
    case 3:
        printf("\nElement at the front is
%d\n",peek());
        break;
    case 4:
        display();
        break;
    case 5:
        exit(1);
    default:
        printf("\nWrong choice\n");
    }
}

return 0;

}

void insert(int item)
{
    if(isFull())
    {
        printf("\nQueue Overflow\n");
        return;
    }
}

```

```

        if(front==-1)
            front=0;
        rear=rear+1;
        queue_arr[rear]=item ;
    }

int del()
{
    int item;
    if(isEmpty())
    {
        printf("\nQueue Underflow\n");
        exit(1);
    }
    item=queue_arr[front];
    front=front+1;
    return item;
}

int peek()
{
    if(isEmpty())
    {
        printf("\nQueue Underflow\n");
        exit(1);
    }
    return queue_arr[front];
}

int isEmpty()
{
    if(front==-1||front==rear+1)
        return 1;
}

```

```

        else

            return 0;

    }

int isFull()
{
    if(rear==MAX-1)
        return 1;
    else
        return 0;
}

void display()
{
    int i;
    if ( isEmpty() )
    {
        printf("\nQueue is empty\n");
        return;
    }
    printf("\nQueue is :\n\n");
    for(i=front;i<=rear;i++)
        printf("%d  ",queue_arr[i]);
    printf("\n\n");
}

```

```

1.Insert
2.Delete
3.Display element at the front
4.Display all elements of the queue
5.Quit

Enter your choice : 1

Input the element for adding in queue : 2
1.Insert
2.Delete
3.Display element at the front
4.Display all elements of the queue
5.Quit

Enter your choice : 1

Input the element for adding in queue : 3
1.Insert
2.Delete
3.Display element at the front
4.Display all elements of the queue
5.Quit

Enter your choice : 4

Queue is :

1  2  3

1.Insert
2.Delete
3.Display element at the front
4.Display all elements of the queue
5.Quit

Enter your choice : 5

```


22. Write a menu driven program of a circular queue using array having add, delete and display operations [without taking global variables

```
#include<stdio.h>

#include<stdlib.h>

#define MAX 10

int cqueue_arr[MAX];

int front=-1;

int rear=-1;

void display( );

void insert(int item);

int del();

int peek();

int isEmpty();

int isFull();

int main()

{

    int choice,item;

    while(1)

    {

        printf("1.Insert\n");

        printf("2.Delete\n");

        printf("3.Peek\n");

        printf("4.Display\n");

        printf("5.Quit\n");

        printf("\nEnter your choice : ");

        scanf("%d",&choice);

        switch(choice)

        {

            case 1 :

                printf("\nInput the element for insertion : ");
```

```

        scanf("%d",&item);
        insert(item);
        break;
    case 2 :
        printf("\nElement deleted is : %d\n",del());
        break;
    case 3:
        printf("\nElement at the front is : 
%d\n",peek());

        break;
    case 4:
        display();
        break;
    case 5:
        exit(1);
    default:
        printf("\nWrong choice\n");
    }

}

return 0;
}

```

```

void insert(int item)
{
    if(isFull())
    {
        printf("\nQueue Overflow\n");
        return;
    }
    if(front== -1)
        front=0;

    if(rear==MAX-1)
        rear=0;
}

```

```

        else
            rear=rear+1;
        cqueue_arr[rear]=item ;
    }

int del()
{
    int item;
    if( isEmpty() )
    {
        printf("\nQueue Underflow\n");
        exit(1);
    }
    item=cqueue_arr[front];
    if(front==rear)
    {
        front=-1;
        rear=-1;
    }
    else if(front==MAX-1)
        front=0;
    else
        front=front+1;
    return item;
}

int isEmpty()
{
    if(front== -1)
        return 1;
    else
        return 0;
}

```

```

int isFull()
{
    if((front==0&&rear==MAX-1) || (front==rear+1))
        return 1;
    else
        return 0;
}

int peek()
{
    if( isEmpty() )
    {
        printf("\nQueue Underflow\n");
        exit(1);
    }
    return cqueue_arr[front];
}

void display()
{
    int i;
    if(isEmpty())
    {
        printf("\nQueue is empty\n");
        return;
    }
    printf("\nQueue elements :\n");
    i=front;
    if( front<=rear )
    {
        while(i<=rear)
            printf("%d ",cqueue_arr[i++]);
    }
}

```

```
    }  
else  
{  
    while(i<=MAX-1)  
        printf("%d ",cqueue_arr[i++]);  
    i=0;  
    while(i<=rear)  
        printf("%d ",cqueue_arr[i++]);  
    }  
    printf("\n");  
}
```

```
1.Insert  
2.Delete  
3.Peek  
4.Display  
5.Quit  
  
Enter your choice : 1  
  
Input the element for insertion : 4  
1.Insert  
2.Delete  
3.Peek  
4.Display  
5.Quit  
  
Enter your choice : █
```

23. Write a menu driven program of an input restricted queue

```
# include<stdio.h>

# define MAX 5

int deque_arr[MAX];

int left = -1;

int right = -1;

void insert_right()
{
    int added_item;

    if((left == 0 && right == MAX-1) || (left == right+1))
    {
        printf("Queue Overflow\n");
        return;}

    if (left == -1)
    {
        left = 0;
        right = 0;}

    else
    if(right == MAX-1)
        right = 0;

    else
        right = right+1;

    printf("Input the element for adding in queue : ");
    scanf("%d", &added_item);
    deque_arr[right] = added_item ;
}

void delete_left()
{
    if (left == -1)
    {
        printf("Queue Underflow\n");
        return ;    }

    printf("Element deleted from queue is : %d\n",deque_arr[left]);
```

```

        if(left == right)
        {
            left = -1;
            right=-1;    }
    else
        if(left == MAX-1)
            left = 0;
        else
            left = left+1;
}

void delete_right()
{if (left == -1)
    {printf("Queue Underflow\n");
        return ;    }
    printf("Element deleted from queue is : %d\n",deque_arr[right]);
    if(left == right)
    {
        left = -1;
        right=-1;    }
    else
        if(right == 0)
            right=MAX-1;
        else
            right=right-1;    }

void display_queue()
{
    int front_pos = left,rear_pos = right;
    if(left == -1)
    {
        printf("Queue is empty\n");
        return;    }
    printf("Queue elements :\n");
    if( front_pos <= rear_pos )
    {
        while(front_pos <= rear_pos)
        {
            printf("%d ",deque_arr[front_pos]);

```

```

        front_pos++;    }    }

else
{
    while(front_pos <= MAX-1)
    {
        printf("%d ",deque_arr[front_pos]);
        front_pos++;    }

    front_pos = 0;
    while(front_pos <= rear_pos)
    {
        printf("%d ",deque_arr[front_pos]);
        front_pos++;
    }
}

printf("\n");
}

void input_que()
{
    int choice;
    do
    {
        printf("1.Insert at right\n");
        printf("2.Delete from left\n");
        printf("3.Delete from right\n");
        printf("4.Display\n");
        printf("5.Quit\n");
        printf("Enter your choice : ");
        scanf("%d",&choice);

        switch(choice)
        {
            case 1:
                insert_right();
                break;

            case 2:
                delete_left();
                break;

            case 3:
                delete_right();

```



```

        break;
    case 4:
        display_queue();
        break;
    case 5:
        break;
    default:
        printf("Wrong choice\n");
    }
}while(choice!=5);
}

int main()
{
    int choice;
    printf("1.Input restricted dequeue\n");
    printf("Enter your choice : ");
    scanf("%d",&choice);
    switch(choice)
    {
        case 1 :
            input_queue();
            break;
        default:
            printf("Wrong choice\n");
    }
}

```

```

1. Insert
2. Delete
3. Display element at the front
4. Display all elements of the queue
5. Quit

Enter your choice : 1

Input the element for adding in queue : 1
1. Insert
2. Delete
3. Display element at the front
4. Display all elements of the queue
5. Quit

Enter your choice : 2

Deleted element is 1
1. Insert
2. Delete
3. Display element at the front

```

24. Write a menu driven program of an output restricted queue.

```
# include<stdio.h>

# define MAX 5

int deque_arr[MAX];
int left = -1;
int right = -1;
void delete_left()
{
    if (left == -1)
    {
        printf("Queue Underflow\n");
        return ;
    }
    printf("Element deleted from queue is : %d\n",deque_arr[left]);
    if(left == right)
    {
        left = -1;
        right=-1;
    }
    else
    {
        if(left == MAX-1)
            left = 0;
        else
            left = left+1;
    }
}

void insert_right()
{
    int added_item;
    if((left == 0 && right == MAX-1) || (left == right+1))
    {
        printf("Queue Overflow\n");
        return;}
    if (left == -1)
    {
        left = 0;
        right = 0;}
    else
    {
        if(right == MAX-1)
```

```

        right = 0;
    else
        right = right+1;
    printf("Input the element for adding in queue : ");
    scanf("%d", &added_item);
    deque_arr[right] = added_item ;
}

```

```

void insert_left()

```

```

{
    int added_item;
    if((left == 0 && right == MAX-1) || (left == right+1))
    {
        printf("Queue Overflow \n");
        return;    }
    if (left == -1)
    {
        left = 0;
        right = 0;    }
    else
    if(left== 0)
        left=MAX-1;
    else
        left=left-1;
    printf("Input the element for adding in queue : ");
    scanf("%d", &added_item);
    deque_arr[left] = added_item ;
}

```

```

void output_que()

```

```

{
    int choice;
    do
    {
        printf("1.Insert at right\n");
        printf("2.Insert at left\n");
        printf("3.Delete from left\n");
        printf("4.Display\n");
    }
}

```

```

        printf("5.Quit\n");
        printf("Enter your choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
                insert_right();
                break;

            case 2:
                insert_left();
                break;

            case 3:
                delete_left();
                break;

            case 4:
                display_queue();
                break;

            case 5:
                break;

            default:
                printf("Wrong choice\n");
        }
    }while(choice!=5);
}

void display_queue()
{
    int front_pos = left,rear_pos = right;
    if(left == -1)
    {
        printf("Queue is empty\n");
        return;    }

    printf("Queue elements :\n");
    if( front_pos <= rear_pos )
    {
        while(front_pos <= rear_pos)
        {
            printf("%d ",deque_arr[front_pos]);

```

```

        front_pos++;    }    }

else
{
    while(front_pos <= MAX-1)
    {
        printf("%d ",deque_arr[front_pos]);
        front_pos++;    }

    front_pos = 0;
    while(front_pos <= rear_pos)
    {
        printf("%d ",deque_arr[front_pos]);
        front_pos++;
    }

}

printf("\n");
}

int main()
{
    int choice;

    printf("1.Output restricted dequeue\n");
    printf("Enter your choice : ");
    scanf("%d",&choice);
    switch(choice)
    {
        case 1:
            output_que();
            break;

        default:
            printf("Wrong choice\n");
    }
}

```

```

1.Output restricted dequeue
Enter your choice : 1
1.Insert at right
2.Insert at left
3.Delete from left
4.Display
5.Quit
Enter your choice : 1
Input the element for adding in queue : 3
1.Insert at right
2.Insert at left
3.Delete from left
4.Display

```

25. Write a program to convert an infix expression to postfix expression.

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

char stack[100];
int top = -1;

void push(char x)
{
    stack[++top] = x;
}

char pop()
{
    if(top == -1)
        return -1;
    else
        return stack[top--];
}

int priority(char x)
{
    if(x == '(')
        return 0;
    if(x == '+' || x == '-')
        return 1;
    if(x == '*' || x == '/')
        return 2;
    return 0;
}

int main()
{
```

```

char exp[100];
char *e, x;
printf("Enter the expression : ");
scanf("%s",exp);
printf("\n");
e = exp;

while(*e != '\0')
{
    if(isalnum(*e))
        printf("%c ",*e);
    else if(*e == '(')
        push(*e);
    else if(*e == ')')
    {
        while((x = pop()) != '(')
            printf("%c ", x);
    }
    else
    {
        while(priority(stack[top]) >= priority(*e))
            printf("%c ",pop());
        push(*e);
    }
    e++;
}

while(top != -1)
{
    printf("%c ",pop());
}return 0;
}

```

Enter the expression : a*b

a b *

PS D:\c prog\mydirectory> █

26. Write a menu driven program of a stack using linked list having the push, pop and display operations[without taking global variables

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

struct node
{
    int info;
    struct node *link;
}*top=NULL;

void push(int item);
int pop();
int peek();
int isEmpty();
void display();

int main()
{
    int choice,item;
    while(1)
    {
        printf("1.Push\n");
        printf("2.Pop\n");
        printf("3.Display item at the top\n");
        printf("4.Display all items of the stack\n");
        printf("5.Quit\n");
        printf("\nEnter your choice : ") ;
        scanf("%d", &choice);

        switch(choice)
        {
            case 1:
                printf("\nEnter the item to be pushed : ");
```



```

        scanf("%d",&item);
        push(item);
        break;
    case 2:
        item=pop();
        printf("\nPopped item is : %d\n",item);
        break;
    case 3:
        printf("\nItem at the top is %d\n",peek());
        break;
    case 4:
        display();
        break;
    case 5:
        exit(1);
    default :
        printf("\nWrong choice\n");
    }
}

```

```

void push(int item)
{
    struct node *tmp;
    tmp=(struct node *)malloc(sizeof(struct node));
    if(tmp==NULL)
    {
        printf("\nStack Overflow\n");
        return;
    }
    tmp->info=item;
    tmp->link=top;
    top=tmp;
}

```

```
}
```

```
int pop()
```

```
{
```

```
    struct node *tmp;
```

```
    int item;
```

```
    if( isEmpty() )
```

```
    {
```

```
        printf("\nStack Underflow\n");
```

```
        exit(1);
```

```
    }
```

```
    tmp=top;
```

```
    item=tmp->info;
```

```
    top=top->link;
```

```
    free(tmp);
```

```
    return item;
```

```
}
```

```
int peek()
```

```
{
```

```
    if( isEmpty() )
```

```
    {
```

```
        printf("\nStack Underflow\n");
```

```
        exit(1);
```

```
    }
```

```
    return top->info;
```

```
}
```

```
int isEmpty()
```

```
{
```

```
    if(top == NULL)
```

```
        return 1;
```

```
    else
```

```

        return 0;
    }

    void display()
    {
        struct node *ptr;
        ptr=top;
        if(isEmpty())
        {
            printf("\nStack is empty\n");
            return;
        }
        printf("\nStack elements :\n\n");
        while(ptr!=NULL)
        {
            printf(" %d\n",ptr->info);
            ptr=ptr->link;
        }
        printf("\n");
    }
}

```

```

1.Push
2.Pop
3.Display item at the top
4.Display all items of the stack
5.Quit

Enter your choice : 1

Enter the item to be pushed : 4
1.Push
2.Pop
3.Display item at the top
4.Display all items of the stack
5.Quit

Enter your choice : 2

Popped item is : 4
1.Push
2.Pop
3.Display item at the top
4.Display all items of the stack
5.Quit

Enter your choice : 3

Stack Underflow

```

27. Write a menu driven program of a queue using linked list having add, delete and display operations [without taking global variables

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

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

```
struct node
```

```
{
```

```
    int info;
```

```
    struct node *link;
```

```
}*front=NULL,*rear=NULL;
```

```
void insert(int item);
```

```
int del();
```

```
int peek();
```

```
int isEmpty();
```

```
void display();
```

```
int main()
```

```
{
```

```
    int choice,item;
```

```
    while(1)
```

```
    {
```

```
        printf("1.Insert\n");
```

```
        printf("2.Delete\n");
```

```
        printf("3.Display the element at the front\n");
```

```
        printf("4.Display all elements of the queue\n");
```

```
        printf("5.Quit\n");
```

```
        printf("\nEnter your choice : ");
```

```
        scanf("%d", &choice);
```

```
        switch(choice)
```

```
        {
```

```
            case 1:
```

```

        printf("\nInput the element for adding in queue
: ");

        scanf("%d",&item);
        insert(item);
        break;

    case 2:
        printf("\nDeleted element is  %d\n",del());
        break;

    case 3:
        printf("\nElement at the front of the queue is
%d\n", peek() );
        break;

    case 4:
        display();
        break;

    case 5:
        exit(1);

    default :
        printf("\nWrong choice\n");

    }

}
}

```

```

void insert(int item)
{
    struct node *tmp;
    tmp=(struct node *)malloc(sizeof(struct node));
    if(tmp==NULL)
    {
        printf("\nMemory not available\n");
        return;
    }
    tmp->info = item;
    tmp->link=NULL;
}

```

```

        if(front==NULL)
            front=tmp;
        else
            rear->link=tmp;
        rear=tmp;
    }

int del()
{
    struct node *tmp;
    int item;
    if( isEmpty( ) )
    {
        printf("\nQueue Underflow\n");
        exit(1);
    }
    tmp=front;
    item=tmp->info;
    front=front->link;
    free(tmp);
    return item;
}

int peek()
{
    if( isEmpty( ) )
    {
        printf("\nQueue Underflow\n");
        exit(1);
    }
    return front->info;
}

```

```

int isEmpty()
{
    if(front==NULL)
        return 1;
    else
        return 0;
}

void display()
{
    struct node *ptr;
    ptr=front;
    if(isEmpty())
    {
        printf("\nQueue is empty\n");
        return;
    }
    printf("\nQueue elements :\n\n");
    while(ptr!=NULL)
    {
        printf("%d ",ptr->info);
        ptr=ptr->link;
    }
    printf("\n\n");
}

```

```

Enter your choice : 1
Input the element for adding in queue
1.Insert
2.Delete
3.Display the element at the front
4.Display all elements of the queue
5.Quit

Enter your choice : 2
Deleted element is 3
1.Insert
2.Delete
3.Display the element at the front
4.Display all elements of the queue
5.Quit

Enter your choice : 4
Queue is empty
1.Insert
2.Delete
3.Display the element at the front
4.Display all elements of the queue
5.Quit

Enter your choice : █

```

28. Write a menu driven program of a circular queue using linked list having add, delete and display operations [without taking global variables]

```
#include<stdio.h>

#include<stdlib.h>

struct node
{
    int data;
    struct node *next;
}*rear=NULL;

void insert(int item);
int del();
void display();
int isEmpty();
int peek();

int main()
{
    int choice,item;
    while(1)
    {
        printf("1.Insert\n");
        printf("2.Delete\n");
        printf("3.Peek\n");
        printf("4.Display\n");
        printf("5.Quit\n");
        printf("\nEnter your choice : ");
        scanf("%d",&choice);

        switch(choice)
        {
            case 1:
```



```

        printf("\nEnter the element for insertion : ");
        scanf("%d",&item);
        insert(item);
        break;
    case 2:
        printf("\nDeleted element is %d\n",del());
        break;
    case 3:
        printf("\nItem at the front of queue is
%d\n",peek());
        break;
    case 4:
        display();
        break;
    case 5:
        exit(1);
    default:
        printf("\nWrong choice\n");
    }
}
}

```

```

void insert(int item)
{
    struct node *temp;
    temp=(struct node *)malloc(sizeof(struct node));
    temp->data=item;
    if(temp==NULL)
    {
        printf("\nMemory not available\n");
        return;
    }

    if( isEmpty() )

```

```

        {
            rear=temp;
            temp->next=rear;
        }
    else
    {
        temp->next=rear->next;
        rear->next=temp;
        rear=temp;
    }
}

del()
{
    int item;
    struct node *temp;
    if( isEmpty() )
    {
        printf("\nQueue underflow\n");
        exit(1);
    }
    if(rear->next==rear)
    {
        temp=rear;
        rear=NULL;
    }
    else
    {
        temp=rear->next;
        rear->next=rear->next->next;
    }
    item=temp->data;
    free(temp);
}

```

```

        return item;
    }

int peek()
{
    if( isEmpty() )
    {
        printf("\nQueue underflow\n");
        exit(1);
    }
    return rear->next->data;
}

int isEmpty()
{
    if( rear == NULL )
        return 1;
    else
        return 0;
}

void display()
{
    struct node *p;
    if(isEmpty())
    {
        printf("\nQueue is empty\n");
        return;
    }
    printf("\nQueue is :\n");
    p=rear->next;
    do
    {

```

```
        printf("%d ",p->data)
    p=p->next;
}while(p!=rear->next);
printf("\n");
}
```

```
1.Insert
2.Delete
3.Peek
4.Display
5.Quit
```

Enter your choice : 1

Enter the element for insertion : 23

```
1.Insert
2.Delete
3.Peek
4.Display
5.Quit
```

Enter your choice : 3

Item at the front of queue is 23

```
1.Insert
2.Delete
3.Peek
4.Display
5.Quit
```

29. Write a menu driven program of a linear linked list with functions: insertion at the beginning, deletion at the end, insertion between two nodes, deletion of first node, deletion of last node, deletion of a node whose position is given, deletion of any node and display[without taking global variables]

```
#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

struct node
{
    int data;
    struct node *next;
}*start=NULL,*q,*t;

void insert_beg();
void insert_end();
int insert_pos();
void display();
void delete_beg();
void delete_end();
int delete_pos();

int main()
{
    int ch;

    while(1)
    {
        printf("\n---- Singly Linked List(SLL) Menu ----");
        printf("\n1.Insert\n2.Display\n3.Delete\n4.Exit\n\n");
        printf("\nEnter your choice(1-4):");
        scanf("%d",&ch);

        switch(ch)
```

```

{
    case 1:
        printf("\n---- Insert Menu ----");
        printf("\n1.Insert at beginning\n2.Insert at
end\n3.Insert at specified position\n4.Exit");
        printf("\nEnter your choice(1-4):");
        scanf("%d",&ch);

        switch(ch)
        {
            case 1: insert_beg();
                    break;
            case 2: insert_end();
                    break;
            case 3: insert_pos();
                    break;
            case 4: exit(0);
            default: printf("Wrong Choice!!");
        }
        break;

    case 2: display();
            break;

    case 3: printf("\n---- Delete Menu ----");
            printf("\n1.Delete from beginning\n2.Delete from
end\n3.Delete from specified position\n4.Exit");
            printf("\nEnter your choice(1-4):");
            scanf("%d",&ch);

            switch(ch)
            {
                case 1: delete_beg();
                        break;

```

```

        case 2: delete_end();
                break;
        case 3: delete_pos();
                break;
        case 4: exit(0);
        default: printf("Wrong Choice!!");
    }
    break;
    case 4: exit(0);
    default: printf("Wrong Choice!!");
}
}
return 0;
}

```

```

void insert_beg()
{
    int num;
    t=(struct node*)malloc(sizeof(struct node));
    printf("Enter data:");
    scanf("%d",&num);
    t->data=num;

    if(start==NULL)
    {
        t->next=NULL;
        start=t;
    }
    else
    {
        t->next=start;
        start=t;
    }
}

```

```
}
```

```
void insert_end()
```

```
{
```

```
    int num;
```

```
    t=(struct node*)malloc(sizeof(struct node));
```

```
    printf("Enter data:");
```

```
    scanf("%d",&num);
```

```
    t->data=num;
```

```
    t->next=NULL;
```

```
    if(start==NULL)
```

```
    {
```

```
        start=t;
```

```
    }
```

```
    else
```

```
    {
```

```
        q=start;
```

```
        while(q->next!=NULL)
```

```
            q=q->next;
```

```
            q->next=t;
```

```
    }
```

```
}
```

```
int insert_pos()
```

```
{
```

```
    int pos,i,num;
```

```
    if(start==NULL)
```

```
    {
```

```
        printf("List is empty!!");
```

```
        return 0;
```

```
    }
```



```

t=(struct node*)malloc(sizeof(struct node));
printf("Enter data:");
scanf("%d",&num);
printf("Enter position to insert:");
scanf("%d",&pos);
t->data=num;

q=start;
for(i=1;i<pos-1;i++)
{
    if(q->next==NULL)
    {
        printf("There are less elements!!");
        return 0;
    }

    q=q->next;
}

t->next=q->next;
q->next=t;
return 0;
}

void display()
{
    if(start==NULL)
    {
        printf("List is empty!!");
    }
    else
    {
        q=start;
        printf("The linked list is:\n");
    }
}

```

```

        while(q!=NULL)
        {
            printf("%d ",q->data);
            q=q->next;
        }
    }
}

```

```

void delete_beg()
{
    if(start==NULL)
    {
        printf("The list is empty!!");
    }
    else
    {
        q=start;
        start=start->next;
        printf("Deleted element is %d",q->data);
        free(q);
    }
}

```

```

void delete_end()
{
    if(start==NULL)
    {
        printf("The list is empty!!");
    }
    else
    {
        q=start;
        while(q->next->next!=NULL)

```

```

        q=q->next;

        t=q->next;
        q->next=NULL;
        printf("Deleted element is %d",t->data);
        free(t);
    }
}

```

```

int delete_pos()
{
    int pos,i;

    if(start==NULL)
    {
        printf("List is empty!!");
        return 0;
    }

    printf("Enter position to delete:");
    scanf("%d",&pos);

    q=start;
    for(i=1;i<pos-1;i++)
    {
        if(q->next==NULL)
        {
            printf("There are less elements!!");
            return 0;
        }
        q=q->next;
    }
}

```

```
t=q->next;

q->next=t->next;

printf("Deleted element is %d",t->data);

free(t);

return 0;

}
```

---- Singly Linked List(SLL) Menu ----

- 1.Insert
- 2.Display
- 3.Delete
- 4.Exit

Enter your choice(1-4):1

---- Insert Menu ----

- 1.Insert at beginning
- 2.Insert at end
- 3.Insert at specified position
- 4.Exit

Enter your choice(1-4):2

Enter data:3

---- Singly Linked List(SLL) Menu ----

- 1.Insert
- 2.Display
- 3.Delete
- 4.Exit

Enter your choice(1-4):

30. Write a menu driven program of a circular linked list with functions: insertion at the beginning, deletion at the end, insertion between two nodes, deletion of first node, deletion of last node, deletion of a node whose position is given, deletion of any node and display

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

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

```
struct node
```

```
{
```

```
    int data;
```

```
    struct node *next;
```

```
};
```

```
struct node *create_list(struct node *last);
```

```
void display(struct node *last);
```

```
struct node *addtoempty(struct node *last,int data);
```

```
struct node *addatbeg(struct node *last,int data);
```

```
struct node *addatend(struct node *last,int data);
```

```
struct node *addafter(struct node *last,int data,int item);
```

```
struct node *del(struct node *last,int data);
```

```
int main( )
```

```
{
```

```
    int choice,data,item;
```

```
    struct node *last=NULL;
```

```
    while(1)
```

```
    {
```

```
        printf("1.Create List\n");
```

```
        printf("2.Display\n");
```

```
        printf("3.Add to empty list\n");
```

```
        printf("4.Add at beginning\n");
```

```
        printf("5.Add at end\n");
```

```
        printf("6.Add after \n");
```

```

printf("7.Delete\n");
printf("8.Quit\n");

printf("\nEnter your choice : ");
scanf("%d",&choice);

switch(choice)
{
    case 1:
        last=create_list(last);
        break;
    case 2:
        display(last);
        break;
    case 3:
        printf("\nEnter the element to be inserted :
");
        scanf("%d",&data);
        last=addtoempty(last,data);
        break;
    case 4:
        printf("\nEnter the element to be inserted :
");
        scanf("%d",&data);
        last=addatbeg(last,data);
        break;
    case 5:
        printf("\nEnter the element to be inserted :
");
        scanf("%d",&data);
        last=addatend(last,data);
        break;
    case 6:
        printf("\nEnter the element to be inserted :
");

```

```

scanf("%d",&data);
printf("\nEnter the element after which to
insert : ");

scanf("%d",&item);
last=addafter(last,data,item);
break;

case 7:
printf("\nEnter the element to be deleted : ");
scanf("%d",&data);
last=del(last,data);
break;

case 8:
exit(1);

default:
printf("\nWrong choice\n");
}

}

return 0;

}

```

```

struct node *create_list(struct node *last)
{
    int i,n,data;
    printf("\nEnter the number of nodes : ");
    scanf("%d",&n);
    last=NULL;
    if(n==0)
        return last;
    printf("Enter the element to be inserted : ");
    scanf("%d",&data);
    last=addtoempty(last,data);
}

```

```

    for(i=2;i<=n;i++)
    {
        printf("Enter the element to be inserted : ");
        scanf("%d",&data);
        last=addatend(last,data);
    }
    return last;
}

```

```

struct node *addtoempty(struct node *last,int data)
{
    struct node *tmp;
    tmp=(struct node *)malloc(sizeof(struct node));
    tmp->data=data;
    last=tmp;
    last->next=last;
    return last;
}

```

```

struct node *addatbeg(struct node *last,int data)
{
    struct node *tmp;
    tmp=(struct node *)malloc(sizeof(struct node));
    tmp->data=data;
    tmp->next=last->next;
    last->next=tmp;
    return last;
}

```

```

struct node *addatend(struct node *last,int data)
{
    struct node *tmp;
    tmp=(struct node *)malloc(sizeof(struct node));

```



```

    tmp->data=data;
    tmp->next=last->next;
    last->next=tmp;
    last=tmp;
    return last;
}

```

```

struct node *addafter(struct node *last,int data,int item)
{
    struct node *tmp,*p;
    p=last->next;
    do
    {
        if(p->data==item)
        {
            tmp=(struct node *)malloc(sizeof(struct node));
            tmp->data=data;
            tmp->next=p->next;
            p->next=tmp;
            if(p==last)
                last=tmp;
            return last;
        }
        p=p->next;
    }while(p!=last->next);
    printf("%d not present in the list\n",item);
    return last;
}

```

```

struct node *del(struct node *last,int data)
{
    struct node *tmp,*p;
    if(last==NULL)

```

```

{
    printf("List is empty\n");
    return last;
}

if(last->next==last && last->data==data)
{
    tmp=last;
    last=NULL;
    free(tmp);
    return last;
}

if(last->next->data==data)
{
    tmp=last->next;
    last->next=tmp->next;
    free(tmp);
    return last;
}

p=last->next;
while(p->next!=last)
{
    if(p->next->data==data)
    {
        tmp=p->next;
        p->next=tmp->next;
        free(tmp);
        return last;
    }
    p=p->next;
}

```

```

        if(last->data==data)
        {
            tmp=last;
            p->next=last->next;
            last=p;
            free(tmp);
            return last;
        }
        printf("\nElement %d not found\n",data);
        return last;
    }

void display(struct node *last)
{
    struct node *p;
    if(last==NULL)
    {
        printf("\nList is empty\n");
        return;
    }
    p=last->next;
    do
    {
        printf("%d ",p->data);
        p=p->next;
    }while(p!=last->next);
    printf("\n");
}

```

```

1.Create List
2.Display
3.Add to empty list
4.Add at beginning
5.Add at end
6.Add after
7.Delete
8.Quit

Enter your choice : 1

Enter the number of nodes : 4
Enter the element to be inserted : 1
Enter the element to be inserted : 2
Enter the element to be inserted : 3
Enter the element to be inserted : 4
1.Create List
2.Display
3.Add to empty list
4.Add at beginning
5.Add at end
6.Add after
7.Delete
8.Quit

Enter your choice : █

```

31. . Write a menu driven program of a doubly linked list with functions: insertion at the beginning, deletion at the end, insertion between two nodes, deletion of first node, deletion of last node, deletion of a node whose position is given, deletion of any node and display[without taking global variables

```
#include <stdio.h>

#include <stdlib.h>

struct node {
    int data;
    struct node *prev, *next;
};

struct node* start = NULL;

void traverse(){

    if (start == NULL) {
        printf("\nList is empty\n");
        return;
    }

    struct node* temp;
    temp = start;
    while (temp != NULL) {
        printf("Data = %d\n", temp->data);
        temp = temp->next;
    }
}

void insertAtFront(){
    int data;
    struct node* temp;
    temp = (struct node*)malloc(sizeof(struct node));
    printf("\nEnter number to be inserted: ");
    scanf("%d", &data);
    temp->data = data;
```

```

temp->prev = NULL;
temp->next = start;
start = temp;
}

```

```

void insertAtEnd(){
    int data;
    struct node *temp, *trav;
    temp = (struct node*)malloc(sizeof(struct node));
    temp->prev = NULL;
    temp->next = NULL;
    printf("\nEnter number to be inserted: ");
    scanf("%d", &data);
    temp->data = data;
    temp->next = NULL;
    trav = start;

    if (start == NULL) {

        start = temp;
    }

    else {
        while (trav->next != NULL)
            trav = trav->next;
        temp->prev = trav;
        trav->next = temp;
    }
}

```

```

void insertAtPosition(){
    int data, pos, i = 1;
    struct node *temp, *newnode;

```

```

newnode = malloc(sizeof(struct node));
newnode->next = NULL;
newnode->prev = NULL;

printf("\nEnter position : ");
scanf("%d", &pos);
printf("\nEnter number to be inserted: ");
scanf("%d", &data);
newnode->data = data;
temp = start;

if (start == NULL) {
    start = newnode;
    newnode->prev = NULL;
    newnode->next = NULL;
}

else if (pos == 1) {
    newnode->next = start;
    newnode->next->prev = newnode;
    newnode->prev = NULL;
    start = newnode;
}

else {
    while (i < pos - 1) {
        temp = temp->next;
        i++;
    }
    newnode->next = temp->next;
    newnode->prev = temp;
    temp->next = newnode;
    temp->next->prev = newnode;
}

```

```

    }
}

void deleteFirst(){
    struct node* temp;
    if (start == NULL)
        printf("\nList is empty\n");
    else {
        temp = start;
        start = start->next;
        if (start != NULL)
            start->prev = NULL;
        free(temp);
    }
}

```

```

void deleteEnd(){
    struct node* temp;
    if (start == NULL)
        printf("\nList is empty\n");
    temp = start;
    while (temp->next != NULL)
        temp = temp->next;
    if (start->next == NULL)
        start = NULL;
    else {
        temp->prev->next = NULL;
        free(temp);
    }
}

```

```

void deletePosition(){
    int pos, i = 1;

```

```

struct node *temp, *position;

temp = start;

if (start == NULL)
    printf("\nList is empty\n");
else {
    printf("\nEnter position : ");
    scanf("%d", &pos);

    if (pos == 1) {
        position = start;
        start = start->next;
        if (start != NULL) {
            start->prev = NULL;
        }
        free(position);
        return;
    }

    while (i < pos - 1)
    {
        temp = temp->next;
        i++;
    }
    position = temp->next;
    if (position->next != NULL)
        position->next->prev = temp;
    temp->next = position->next;
    free(position);
}
}

int main(){

```



```
int choice;
while (1) {

    printf("1.To see list\n");
    printf("2.For insertion at "
           " starting\n");
    printf("3.For insertion at "
           " end\n");
    printf("4.For insertion at "
           "any position\n");
    printf("5.For deletion of "
           "first element\n");
    printf("6.For deletion of "
           "last element\n");
    printf("7.For deletion of "
           "element at any position\n");
    printf("8.To exit\n");
    printf("\nEnter Choice :\n");
    scanf("%d", &choice);

    switch (choice) {
    case 1:
        traverse();
        break;
    case 2:
        insertAtFront();
        break;
    case 3:
        insertAtEnd();
        break;
    case 4:
        insertAtPosition();
        break;
```

```

case 5:
    deleteFirst();
    break;
case 6:
    deleteEnd();
    break;
case 7:
    deletePosition();
    break;

case 8:
    exit(1);
    break;
default:
    printf("Incorrect Choice. Try Again \n");
    continue;
}
}
return 0;
}

```

```

1.To see list
2.For insertion at starting
3.For insertion at end
4.For insertion at any position
5.For deletion of first element
6.For deletion of last element
7.For deletion of element at any position
8.To exit

Enter Choice :
2

Enter number to be inserted: 3
1.To see list
2.For insertion at starting
3.For insertion at end
4.For insertion at any position
5.For deletion of first element
6.For deletion of last element
7.For deletion of element at any position
8.To exit

Enter Choice :
1
Data = 3
1.To see list
2.For insertion at starting

```

32. Write a menu driven program of a linear linked list with a header having functions: insertion at the beginning, deletion at the end, insertion between two nodes, deletion of first node, deletion of last node, deletion of a node whose position is given, deletion of any node and display.

```
#include<stdio.h>

#include<conio.h>

#include<process.h>

#include<stdlib.h>

struct node

{

    int data;

    struct node *next;

}*start=NULL,*q,*t;

int main()

{

    int ch;

    void insert_beg();

    void insert_end();

    int insert_pos();

    void display();

    void delete_beg();

    void delete_end();

    int delete_pos();

    while(1)

    {

        printf("\n\n---- Singly Linked List(SLL) Menu ----");

        printf("\n1.Insert\n2.Display\n3.Delete\n4.Exit\n\n");

        printf("Enter your choice(1-4):");

        scanf("%d",&ch);

        switch(ch)

        {
```

```

case 1:

    printf("\n---- Insert Menu ----");

    printf("\n1.Insert at beginning\n2.Insert at
end\n3.Insert at specified position\n4.Exit");

    printf("\n\nEnter your choice(1-4):");
    scanf("%d",&ch);

    switch(ch)
    {
        case 1: insert_beg();
                break;
        case 2: insert_end();
                break;
        case 3: insert_pos();
                break;
        case 4: exit(0);
        default: printf("Wrong Choice!!");
    }
    break;

case 2: display();
        break;

case 3: printf("\n---- Delete Menu ----");

        printf("\n1.Delete from beginning\n2.Delete from
end\n3.Delete from specified position\n4.Exit");

        printf("\n\nEnter your choice(1-4):");
        scanf("%d",&ch);

        switch(ch)
        {
            case 1: delete_beg();
                    break;
            case 2: delete_end();

```

```

                break;
            case 3: delete_pos();
                break;
            case 4: exit(0);
            default: printf("Wrong Choice!!");
        }
        break;
    case 4: exit(0);
    default: printf("Wrong Choice!!");
}
}
return 0;
}

void insert_beg()
{
    int num;
    t=(struct node*)malloc(sizeof(struct node));
    printf("Enter data:");
    scanf("%d",&num);
    t->data=num;

    if(start==NULL)
    {
        t->next=NULL;
        start=t;
    }
    else
    {
        t->next=start;
        start=t;
    }
}

```

```

void insert_end()
{
    int num;
    t=(struct node*)malloc(sizeof(struct node));
    printf("Enter data:");
    scanf("%d",&num);
    t->data=num;
    t->next=NULL;

    if(start==NULL)
    {
        start=t;
    }
    else
    {
        q=start;
        while(q->next!=NULL)
        {
            q=q->next;
        }
        q->next=t;
    }
}

int insert_pos()
{
    int pos,i,num;
    if(start==NULL)
    {
        printf("List is empty!!");
        return 0;
    }

    t=(struct node*)malloc(sizeof(struct node));

```

```

printf("Enter data:");
scanf("%d",&num);
printf("Enter position to insert:");
scanf("%d",&pos);
t->data=num;

q=start;
for(i=1;i<pos-1;i++)
{
    if(q->next==NULL)
    {
        printf("There are less elements!!");
        return 0;
    }

    q=q->next;
}

t->next=q->next;
q->next=t;
return 0;
}

void display()
{
    if(start==NULL)
    {
        printf("List is empty!!");
    }
    else
    {
        q=start;
        printf("The linked list is:\n");
    }
}

```

```

        while(q!=NULL)
        {
            printf("%d->", q->data);
            q=q->next;
        }
    }
}

```

```

void delete_beg()
{
    if(start==NULL)
    {
        printf("The list is empty!!");
    }
    else
    {
        q=start;
        start=start->next;
        printf("Deleted element is %d", q->data);
        free(q);
    }
}

```

```

void delete_end()
{
    if(start==NULL)
    {
        printf("The list is empty!!");
    }
    else
    {
        q=start;
        while(q->next->next!=NULL)

```



```

        q=q->next;

        t=q->next;
        q->next=NULL;
        printf("Deleted element is %d",t->data);
        free(t);
    }
}

```

```

int delete_pos()
{
    int pos,i;

    if(start==NULL)
    {
        printf("List is empty!!");
        return 0;
    }

    printf("Enter position to delete:");
    scanf("%d",&pos);

    q=start;
    for(i=1;i<pos-1;i++)
    {
        if(q->next==NULL)
        {
            printf("There are less elements!!");
            return 0;
        }
        q=q->next;
    }
}

```

```
t=q->next;

q->next=t->next;

printf("Deleted element is %d",t->data);

free(t);

return 0;
}
```

```
---- Singly Linked List(SLL) Menu ----
1.Insert
2.Display
3.Delete
4.Exit
```

Enter your choice(1-4):1

```
---- Insert Menu ----
1.Insert at beginning
2.Insert at end
3.Insert at specified position
4.Exit
```

Enter your choice(1-4):1
Enter data:3

```
---- Singly Linked List(SLL) Menu ----
1.Insert
2.Display
3.Delete
4.Exit
```

Enter your choice(1-4):2
The linked list is:
3->

33. Write a program to create a linked list P, then write a 'C' function named split to create two linked lists Q & R from P so that Q contains all elements in odd positions of P and R contains the remaining elements. Finally print both linked lists i.e. Q and R

```
#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

typedef struct node
{
    int data;
    struct node *next;
}s;

s*even=NULL;
s*odd=NULL;

void insertion(s **,s**,int);
void arrange(s*);
void display(s*,s*);
void main()
{
    s*f=NULL;
    s*r=NULL;
    int n,ch;
    do
    {
        printf("\npress 1 to enter node \n2 to display all
nodes\n3.arrange\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:
                printf("Enter the number:");
                scanf("%d",&n);
                insertion(&f,&r,n);
                break;
            case 2:
```

```

        display(even, odd);

        break;

        case 3:
            arrange(f);
    }

    }while(ch!=4);

    getch();
}

void insertion(s**first,s**last,int n)
{
    s* p=NULL;
    p=((s*)malloc(sizeof(s)));
    p->data=n;
    p->next=NULL;
    if ((*last)==NULL)
    {
        (*last)=p;
        (*first)=(*last);
    }
    else
    {
        (*last)->next=p;
        (*last)=(*last)->next;
    }
}

void arrange(s*f)
{
    s*r=NULL;
    s*q=NULL;
    s*g=NULL;
    int t;
    q=f;
    while(q!=NULL)

```

```

{
    t=q->data;
    if((t%2)==0)
    {
        r=(s*)malloc(sizeof(s));
        r->data=t;
        r->next=NULL;
        if(even==NULL)
        {
            even=r;
        }
        else
        {
            r->next=even;
            even=r;
        }
    }
    else
    {
        g=(s*)malloc(sizeof(s));
        g->data=t;
        g->next=NULL;
        if(odd==NULL)
        {
            odd=r;
        }
        else
        {
            g->next=odd;
            odd=g;
        }
    }
    q=q->next;
}

```

```

    }
}
void display(s* even,s* odd)
{
    s*b=NULL;
    s*x=NULL;
    b=even;
    x=odd;
    printf("The output of even link list is: \n");
    while(b!=NULL)
    {
        printf("%d\t",b->data);
        b=b->next;
    }
    printf("\nThe output of odd link list is: \n");
    while(x!=NULL)
    {
        printf("%d\t",x->data);
        x=x->next;
    }
}

```

```

press 1 to enter node
2 to display all nodes
3.arrange
1
Enter the number:3

```

```

press 1 to enter node
2 to display all nodes
3.arrange

```

34. Write a program to sort and merge linked lists.

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

struct node
{
    int data;
    struct node *next;
};

struct node *create(struct node *start);
struct node *insert_s(struct node *start,int data);
struct node *insert(struct node *start,int data);
void display(struct node *start );
void merge(struct node *p1,struct node *p2);
int main()
{
    struct node *start1=NULL,*start2=NULL;
    start1=create(start1);
    start2=create(start2);

    printf("List1 : ");
    display(start1);
    printf("List2 : ");
    display(start2);
    merge(start1, start2);

    return 0;
}

void merge(struct node *p1,struct node *p2)
{

```

```

struct node *start3;

start3=NULL;

while(p1!=NULL && p2!=NULL)
{
    if(p1->data < p2->data)
    {
        start3=insert(start3,p1->data);
        p1=p1->next;
    }
    else if(p2->data < p1->data)
    {
        start3=insert(start3,p2->data);
        p2=p2->next;
    }
    else if(p1->data==p2->data)
    {
        start3=insert(start3,p1->data);
        p1=p1->next;
        p2=p2->next;
    }
}

while(p1!=NULL)
{
    start3=insert(start3,p1->data);
    p1=p1->next;
}

while(p2!=NULL)
{
    start3=insert(start3,p2->data);
    p2=p2->next;
}

```



```

        printf("Merged list is : ");
        display(start3);
    }

struct node *create(struct node *start )
{
    int i,n,data;
    printf("Enter the number of nodes : ");
    scanf("%d",&n);
    start=NULL;
    for(i=1;i<=n;i++)
    {
        printf("Enter the element to be inserted : ");
        scanf("%d",&data);
        start=insert_s(start, data);
    }
    return start;
}

struct node *insert_s(struct node *start,int data)
{
    struct node *p,*temp;
    temp=(struct node *)malloc(sizeof(struct node));
    temp->data=data;
    if(start==NULL || data<start->data)
    {
        temp->next=start;
        start=temp;
        return start;
    }
    else
    {
        p=start;

```

```

        while(p->next!=NULL && p->next->data < data)
            p=p->next;
        temp->next=p->next;
        p->next=temp;
    }
    return start;
}

```

```

struct node *insert(struct node *start,int data)
{
    struct node *p,*temp;
    temp=(struct node *)malloc(sizeof(struct node));
    temp->data=data;
    if(start==NULL)
    {
        temp->next=start;
        start=temp;
        return start;
    }
    else
    {
        p=start;
        while(p->next!=NULL)
            p=p->next;
        temp->next=p->next;
        p->next=temp;
    }
    return start;
}

```

```

void display(struct node *start)
{
    struct node *p;

```

```

if(start==NULL)
{
    printf("List is empty\n");
    return;
}
p=start;
while(p!=NULL)
{
    printf("%d ",p->data);
    p=p->next;
}
printf("\n");
}

```

```

59 } ; 11 ($?) { .\59 }
Enter the number of nodes : 2
Enter the element to be inserted : 1
Enter the element to be inserted : 2
Enter the number of nodes : 2
Enter the element to be inserted : 3
Enter the element to be inserted : 4
List1 : 1 2
List2 : 3 4
Merged list is : 1 2 3 4
59 } ; 11 ($?) { .\59 }

```

35 Write a program of a sorted linked list

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

struct node {
    int data;
    struct node *prev, *next;
};

struct node* start = NULL;

void traverse(){

    if (start == NULL) {
        printf("\nList is empty\n");
        return;
    }

    struct node* temp;
    temp = start;
    while (temp != NULL) {
        printf("Data = %d\n", temp->data);
        temp = temp->next;
    }
}

void insertAtFront(){
    int data;
    struct node* temp;
    temp = (struct node*)malloc(sizeof(struct node));
    printf("\nEnter number to be inserted: ");
    scanf("%d", &data);
    temp->data = data;
    temp->prev = NULL;
    temp->next = start;
```

```

        start = temp;
    }

void insertAtEnd(){
    int data;
    struct node *temp, *trav;
    temp = (struct node*)malloc(sizeof(struct node));
    temp->prev = NULL;
    temp->next = NULL;
    printf("\nEnter number to be inserted: ");
    scanf("%d", &data);
    temp->data = data;
    temp->next = NULL;
    trav = start;

    if (start == NULL) {

        start = temp;
    }

    else {
        while (trav->next != NULL)
            trav = trav->next;
        temp->prev = trav;
        trav->next = temp;
    }
}

void insertAtPosition(){
    int data, pos, i = 1;
    struct node *temp, *newnode;
    newnode = malloc(sizeof(struct node));
    newnode->next = NULL;

```

```

newnode->prev = NULL;

printf("\nEnter position : ");
scanf("%d", &pos);
printf("\nEnter number to be inserted: ");
scanf("%d", &data);
newnode->data = data;
temp = start;

if (start == NULL) {
    start = newnode;
    newnode->prev = NULL;
    newnode->next = NULL;
}

else if (pos == 1) {
    newnode->next = start;
    newnode->next->prev = newnode;
    newnode->prev = NULL;
    start = newnode;
}

else {
    while (i < pos - 1) {
        temp = temp->next;
        i++;
    }
    newnode->next = temp->next;
    newnode->prev = temp;
    temp->next = newnode;
    temp->next->prev = newnode;
}
}

```

```

void deleteFirst(){
    struct node* temp;
    if (start == NULL)
        printf("\nList is empty\n");
    else {
        temp = start;
        start = start->next;
        if (start != NULL)
            start->prev = NULL;
        free(temp);
    }
}

```

```

void deleteEnd(){
    struct node* temp;
    if (start == NULL)
        printf("\nList is empty\n");
    temp = start;
    while (temp->next != NULL)
        temp = temp->next;
    if (start->next == NULL)
        start = NULL;
    else {
        temp->prev->next = NULL;
        free(temp);
    }
}

```

```

void deletePosition(){
    int pos, i = 1;
    struct node *temp, *position;
    temp = start;

```

```

    if (start == NULL)
        printf("\nList is empty\n");
    else {
        printf("\nEnter position : ");
        scanf("%d", &pos);

        if (pos == 1) {
            position = start;
            start = start->next;
            if (start != NULL) {
                start->prev = NULL;
            }
            free(position);
            return;
        }

        while (i < pos - 1)
        {
            temp = temp->next;
            i++;
        }
        position = temp->next;
        if (position->next != NULL)
            position->next->prev = temp;
        temp->next = position->next;
        free(position);
    }
}

int main(){
    int choice;
    while (1) {

```



```
printf("1.To see list\n");
printf("2.For insertion at"
      " starting\n");
printf("3.For insertion at"
      " end\n");
printf("4.For insertion at "
      "any position\n");
printf("5.For deletion of "
      "first element\n");
printf("6.For deletion of "
      "last element\n");
printf("7.For deletion of "
      "element at any position\n");
printf("8.To exit\n");
printf("\nEnter Choice :\n");
scanf("%d", &choice);

switch (choice) {
case 1:
    traverse();
    break;
case 2:
    insertAtFront();
    break;
case 3:
    insertAtEnd();
    break;
case 4:
    insertAtPosition();
    break;
case 5:
    deleteFirst();
```

```

        break;
case 6:
    deleteEnd();
    break;
case 7:
    deletePosition();
    break;

case 8:
    exit(1);
    break;
default:
    printf("Incorrect Choice. Try Again \n");
    continue;
}
}
return 0;
}

```

```

1.To see list
2.For insertion at starting
3.For insertion at end
4.For insertion at any position
5.For deletion of first element
6.For deletion of last element
7.For deletion of element at any position
8.To exit

Enter Choice :
2

Enter number to be inserted: 3
1.To see list
2.For insertion at starting
3.For insertion at end
4.For insertion at any position
5.For deletion of first element
6.For deletion of last element
7.For deletion of element at any position
8.To exit

Enter Choice :
1
Data = 3
1.To see list
2.For insertion at starting
3.For insertion at end
4.For insertion at any position
5.For deletion of first element
6.For deletion of last element
7.For deletion of element at any position
8.To exit

Enter Choice :

```

36. Write a program to add two polynomial

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

typedef struct link {
    int coeff;
    int pow;
    struct link * next;
} my_poly;

void my_create_poly(my_poly **);
void my_show_poly(my_poly *);
void my_add_poly(my_poly **, my_poly *, my_poly *);

int main(void) {
    int ch;
    do {
        my_poly * poly1, * poly2, * poly3;

        printf("\nCreate 1st expression\n");
        my_create_poly(&poly1);
        printf("\nStored the 1st expression");
        my_show_poly(poly1);

        printf("\nCreate 2nd expression\n");
        my_create_poly(&poly2);
        printf("\nStored the 2nd expression");
        my_show_poly(poly2);

        my_add_poly(&poly3, poly1, poly2);
        my_show_poly(poly3);

        printf("\nAdd two more expressions? (Y = 1/N = 0): ");
```

```

        scanf("%d", &ch);
    } while (ch);
    return 0;
}

void my_create_poly(my_poly ** node) {
    int flag;
    int coeff, pow;
    my_poly * tmp_node;
    tmp_node = (my_poly *) malloc(sizeof(my_poly));
    *node = tmp_node;
    do {

        printf("\nEnter Coeff:");
        scanf("%d", &coeff);
        tmp_node->coeff = coeff;
        printf("\nEnter Pow:");
        scanf("%d", &pow);
        tmp_node->pow = pow;
        tmp_node->next = NULL;

        printf("\nContinue adding more terms to the polynomial list?(Y
= 1/N = 0): ");
        scanf("%d", &flag);

        if(flag) {
            tmp_node->next = (my_poly *) malloc(sizeof(my_poly));
            tmp_node = tmp_node->next;
            tmp_node->next = NULL;
        }
    } while (flag);
}

void my_show_poly(my_poly * node) {

```

```

printf("\nThe polynomial expression is:\n");
while(node != NULL) {
    printf("%dx^%d", node->coeff, node->pow);
    node = node->next;
    if(node != NULL)
        printf(" + ");
}
}

void my_add_poly(my_poly ** result, my_poly * poly1, my_poly * poly2) {
    my_poly * tmp_node;
    tmp_node = (my_poly *) malloc(sizeof(my_poly));
    tmp_node->next = NULL;
    *result = tmp_node;

    while(poly1 && poly2) {
        if (poly1->pow > poly2->pow) {
            tmp_node->pow = poly1->pow;
            tmp_node->coeff = poly1->coeff;
            poly1 = poly1->next;
        }
        else if (poly1->pow < poly2->pow) {
            tmp_node->pow = poly2->pow;
            tmp_node->coeff = poly2->coeff;
            poly2 = poly2->next;
        }
        else {
            tmp_node->pow = poly1->pow;
            tmp_node->coeff = poly1->coeff + poly2->coeff;
            poly1 = poly1->next;
            poly2 = poly2->next;
        }
    }
}

```

```

        if(poly1 && poly2) {
            tmp_node->next = (my_poly *) malloc(sizeof(my_poly));
            tmp_node = tmp_node->next;
            tmp_node->next = NULL;
        }
    }

while(poly1 || poly2)
{
    tmp_node->next = (my_poly *) malloc(sizeof(my_poly));
    tmp_node = tmp_node->next;
    tmp_node->next = NULL;

    if(poly1) {
        tmp_node->pow = poly1->pow;
        tmp_node->coeff = poly1->coeff;
        poly1 = poly1->next;
    }

    if(poly2) {
        tmp_node->pow = poly2->pow;
        tmp_node->coeff = poly2->coeff;
        poly2 = poly2->next;
    }
}

printf("\nAddition Complete");
}

```

Create 1st expression

Enter Coeff:4

Enter Pow:2

Continue adding more terms to the polynomial list?(Y = 1/N = 0): 1

Enter Coeff:4

Enter Pow:2

Continue adding more terms to the polynomial list?(Y = 1/N = 0):

37. . Write a program to implement ascending priority queue with following functions i)insert ii)serve iii)display (use double pointer)

```
#include<stdio.h>
#include<conio.h>
#include<malloc.h>
typedef struct que
{
    int info,prio;
    struct que *next;
}nodetype;
void insert(nodetype**,nodetype**,int,int);
void serve(nodetype**,nodetype**);
void display(nodetype*);
void main()
{
    nodetype *front=NULL,*rear=NULL;
    int ch,pr,num,c=0;
    printf("Menu\n1.Insert\n2.serve\n3.Display\n4.Exit\nChoice : ");
    do
    {
        scanf("%d",&ch);
        switch(ch)
        {
            case 1: printf("\nEnter the number to be inserted and its
priority : ");
                scanf("%d %d",&num,&pr);
                insert(&front,&rear,num,pr);
                break;
            case 2: if(front==NULL)
                printf("\nNo elements are present in the Priority Queue");
                else
                serve(&front,&rear);
                break;
            case 3: if(front==NULL)
```

```

        printf("\nNo elements in the Priority Queue");
        else
            display(front);
        break;
        default:
            printf("\nEnter the correct choice");
    }
    printf("press 1 to continue\n");
    scanf("%d",&c);
}while(c==1);
}

void insert(nodetype **front,nodetype **rear,int num,int pr)
{
    nodetype *p=NULL,*tmp=*front;
    p=(nodetype*)malloc(sizeof(nodetype));
    if(p!=NULL)
    {
        p->info=num;
        p->prio=pr;
        p->next=NULL;
    }
    if((*rear)==NULL)
    {
        (*front)=(*rear)=p;
    }
    else
    {
        if((p->prio)>((*rear)->prio))
        {
            (*rear)->next=p;
            (*rear)=p;
        }
        else if(p->prio<((*front)->prio))

```



```

        {
            p->next=(*front);
            (*front)=p;
        }
    else
    {
        while(((tmp->next)->prio)<(p->prio))
            tmp=tmp->next;
        p->next=tmp->next;
        tmp->next=p;
    }
}

}

void serve(nodetype **front,nodetype **rear)
{
    nodetype *tmp=*front;
    if((*front)==(*rear))
    {
        printf("The served element is %d",(*front)->info);
        (*rear)=(*front)=NULL;
    }
    else
    {
        printf("The served element is %d",(*front)->info);
        (*front)=(*front)->next;
    }
    free(tmp);
}

void display(nodetype *front)
{
    nodetype *tmp;
    tmp=front;
    while(tmp!=NULL)

```

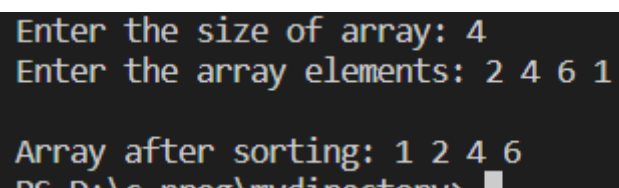
```
{  
    printf("%d ", tmp->info);  
    tmp=tmp->next;  
}  
}
```

```
Menu  
1.Insert  
2.serve  
3.Display  
4.Exit  
Choice : 1  
  
Enter the number to be inserted and its priority : 4  
2  
press 1 to continue  
1
```

38. Write a program to sort a given array using bubble sort.

```
#include<stdio.h>

int main()
{
    int a[50],n,i,j,temp;
    printf("Enter the size of array: ");
    scanf("%d",&n);
    printf("Enter the array elements: ");
    for(i=0;i<n;++i)
        scanf("%d",&a[i]);
    for(i=1;i<n;++i)
        for(j=0;j<(n-i);++j)
            if(a[j]>a[j+1])
            {
                temp=a[j];
                a[j]=a[j+1];
                a[j+1]=temp;
            }
    printf("\nArray after sorting: ");
    for(i=0;i<n;++i)
        printf("%d ",a[i]);
    return 0;
}
```

A screenshot of a terminal window showing the execution of the bubble sort program. The text is displayed in a monospaced font on a dark background. The output shows the user entering the size of the array as 4, then the array elements 2, 4, 6, and 1. Finally, it displays the sorted array: 1, 2, 4, 6.

```
Enter the size of array: 4
Enter the array elements: 2 4 6 1

Array after sorting: 1 2 4 6
```

39. Write a program to sort a given array using selection sort.

```
#include <stdio.h>

int main()
{
    int a[100],n,i,j,temp;
    printf("\n Please Enter the total n of Elements  :  ");
    scanf("%d", &n);
    printf("\n Please Enter the Array Elements  :  ");
    for(i=0;i<n;i++)
        scanf("%d",&a[i]);

    for(i=0;i<n;i++)
    {
        for(j=i+1;j<n;j++)
        {
            if(a[i]>a[j])
            {
                temp=a[i];
                a[i]=a[j];
                a[j]=temp;
            }
        }
    }

    printf("\nSelection Sort Result : ");
    for(i=0;i<n;i++)
    {
        printf("%d\t",a[i]);
    }

    printf("\n");
    return 0;
}
```

Please Enter the total n of Elements : 4

Please Enter the Array Elements : 3 2 6 4

Selection Sort Result : 2 3 4 6

40. Write a program to sort a given array using insertion sort.

```
#include<stdio.h>

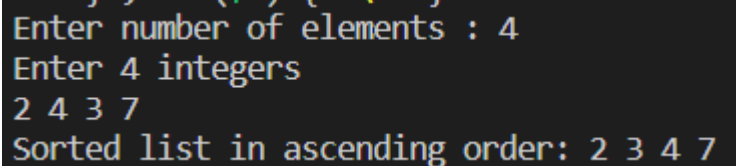
int main()
{
    int n,a[1000],c,d,temp;
    printf("Enter number of elements : ");
    scanf("%d",&n);
    printf("Enter %d integers\n", n);

    for (c=0;c<n;c++)
    {
        scanf("%d", &a[c]);
    }
    for (c=1;c<=n-1;c++) {
        d=c;
        while (d>0&& a[d]<a[d-1]) {
            temp=a[d];
            a[d]=a[d-1];
            a[d-1]=temp;
            d--;
        }
    }

    printf("Sorted list in ascending order:\t");

    for (c=0;c<=n-1;c++)
    {
        printf("%d ",a[c]);
    }

    return 0;
}
```

A screenshot of a terminal window showing the execution of the insertion sort program. The user enters 4 for the number of elements and 2 4 3 7 for the integers. The output shows the sorted list in ascending order: 2 3 4 7.

```
Enter number of elements : 4
Enter 4 integers
2 4 3 7
Sorted list in ascending order: 2 3 4 7
```

41. . Write a program to sort a given array using merge sort

```
#include <stdio.h>

#define max 10

int a[11]={10,14,19,26,27,31,33,35,42,44,0};
int b[10];

void merging(int s, int mid, int e) {
    int l1,l2,i;

    for(l1=s,l2=mid +1,i=s;l1<=mid&&l2<=e;i++)
    {
        if(a[l1]<=a[l2])
            b[i]=a[l1++];
        else
            b[i]=a[l2++];
    }

    while(l1<=mid)
        b[i++]=a[l1++];

    while(l2<=e)
        b[i++]=a[l2++];

    for(i=s;i<=e;i++)
        a[i]=b[i];
}

void sort(int s,int e) {
    int mid;

    if(s < e)
    {
        mid = s+(e-s)/2;
        sort(s,mid);
```

```

        sort(mid+1,e);
        merging(s,mid,e);
    }
else
{
    return;
}
}

int main() {
    int i;
    printf("List before sorting\n");
    for(i=0;i<=max;i++)
        printf("%d ",a[i]);

    sort(0,max);

    printf("\nList after sorting\n");

    for(i=0;i<=max;i++)
        printf("%d ",a[i]);
}

```

```

.59 }
List before sorting
10 14 19 26 27 31 33 35 42 44 0
List after sorting
0 10 14 19 26 27 31 33 35 42 44
PS D:\c prog\mydirectory>

```

42. Write a program to sort a given array using quick sort

```
#include <stdio.h>

int main()
{
    int c, s,e,mid,n,key,a[100];

    printf("Enter number of elements\n");
    scanf("%d",&n);

    printf("Enter %d integers\n",n);

    for (c=0;c<n;c++)
        scanf("%d",&a[c]);

    printf("Enter value to find\n");
    scanf("%d",&key);

    s=0;
    e=n- 1;
    mid=s+(e-s)/2;
    while(s<=e)
    {
        if (a[mid]<key)
            s= mid+1;
        else if (a[mid]==key)
        {
            printf("%d found at location %d.\n", key, mid+1);
            break;
        }
        else
            e=mid-1;
        mid=s+(e-s)/2;
    }
```



```
    if (s>e)
        printf("Not found! %d isn't present in the list.\n", key);
    return 0;
}
```

Enter number of elements

5

Enter 5 integers

2 6 4 8 9

Enter value to find

3

Not found! 3 isn't present in the list.

PS D:\c prog\mydirectory> █

43. Write a program to implement binary search.

```
#include <stdio.h>

int main()
{
    int c, s,e,mid,n,key,a[100];

    printf("Enter number of elements\n");
    scanf("%d",&n);

    printf("Enter %d integers\n",n);

    for (c=0;c<n;c++)
        scanf("%d",&a[c]);

    printf("Enter value to find\n");
    scanf("%d",&key);

    s=0;
    e=n- 1;
    mid=s+(e-s)/2;
    while(s<=e)
    {
        if (a[mid]<key)
            s= mid+1;
        else if (a[mid]==key)
        {
            printf("%d found at location %d.\n", key, mid+1);
            break;
        }
        else
            e=mid-1;
        mid=s+(e-s)/2;
    }
```

```
if (s>e)
    printf("Not found! %d isn't present in the list.\n", key);
return 0;
}
```

```
.\59 }
Enter number of elements
4
Enter 4 integers
1 2 3 5
Enter value to find
4
Not found! 4 isn't present in the list.
```

44. Write a program to implement a binary search tree with following operation: insertion, deletion and display

```
#include <stdio.h>

#include <stdlib.h>

struct node {
    int key;
    struct node *left, *right;
};

struct node *newNode(int item) {
    struct node *temp = (struct node *)malloc(sizeof(struct node));
    temp->key = item;
    temp->left = temp->right = NULL;
    return temp;
}

void inorder(struct node *root) {
    if (root != NULL)
    {
        inorder(root->left);
        printf("%d -> ", root->key);
        inorder(root->right);
    }
}

struct node *insert(struct node *node, int key) {
    if (node == NULL)
        return newNode(key);

    if (key < node->key)
        node->left = insert(node->left, key);
    else
```

```

    node->right = insert(node->right, key);

    return node;
}

struct node *minValueNode(struct node *node) {
    struct node *current = node;
    while (current && current->left != NULL)
        current = current->left;

    return current;
}

struct node *deleteNode(struct node *root, int key) {
    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) {
            struct node *temp = root->right;
            free(root);
            return temp;
        } else if (root->right == NULL) {
            struct node *temp = root->left;
            free(root);
            return temp;
        }

        struct node *temp = minValueNode(root->right);

```

```

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

```

```

int main() {
    struct node *root = NULL;
    root = insert(root, 8);
    root = insert(root, 3);
    root = insert(root, 1);
    root = insert(root, 6);
    root = insert(root, 7);
    root = insert(root, 10);
    root = insert(root, 14);
    root = insert(root, 4);

    printf("Inorder traversal: ");
    inorder(root);

    printf("\nAfter deleting 10\n");
    root = deleteNode(root, 10);
    printf("Inorder traversal: ");
    inorder(root);
}

```

```

.59 }
Inorder traversal: 1 -> 3 -> 4 -> 6 -> 7 -> 8 -> 10 -> 14 ->
After deleting 10
Inorder traversal: 1 -> 3 -> 4 -> 6 -> 7 -> 8 -> 14 ->

```

45. Program to search a key and delete particular node from a binary search tree & count no. of node, leaf node & height of tree.

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

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

```
typedef struct TreeNode
```

```
{
```

```
    int data;
```

```
    struct TreeNode *left;
```

```
    struct TreeNode *right;
```

```
} TreeNode;
```

```
TreeNode *root = NULL;
```

```
void find(int x, TreeNode **loc, TreeNode **par)
```

```
{
```

```
    TreeNode *ptr, *ptrpar;
```

```
    if (root == NULL)
```

```
    {
```

```
        *par = NULL;
```

```
        *loc = NULL;
```

```
        return;
```

```
    }
```

```
    if (x == root->data)
```

```
    {
```

```
        *par = NULL;
```

```
        *loc = root;
```

```
        return;
```

```
    }
```

```
    ptrpar = root;
```

```
    if (x < root->data)
```

```
        ptr = root->left;
```

```
    if (x > root->data)
```

```
        ptr = root->right;
```

```

while (ptr != NULL)
{
    if (x == ptr->data)
    {
        *loc = ptr;
        *par = ptrpar;
        return;
    }
    ptrpar = ptr;
    if (x < ptr->data)
        ptr = ptr->left;
    else if (x > ptr->data)
        ptr = ptr->right;
}
*par = ptrpar;
*loc = ptr;
}

void insert(int x)
{
    TreeNode *newN = (TreeNode *)malloc(sizeof(TreeNode));
    if (newN == NULL)
    {
        printf("Overflow\n");
        return;
    }
    newN->data = x;
    newN->left = NULL;
    newN->right = NULL;
    if (root == NULL)
    {
        root = newN;
        return;
    }
}

```



```

    }

    TreeNode *loc, *par;
    find(x, &loc, &par);
    if (loc != NULL)
    {
        printf("Exists\n");
        return;
    }

    if (x < par->data)
        par->left = newN;

    else if (x > par->data)
        par->right = newN;
}

void Print(TreeNode *temp)
{
    if (temp == NULL)
    {
        return;
    }
    // LNR
    Print(temp->left);
    printf("%d ", temp->data);
    Print(temp->right);
}

void case1(TreeNode *loc, TreeNode *par)
{
    if (par == NULL)

```

```

    {
        root = NULL;
    }
    if (par->left == loc)
    {
        par->left = NULL;
    }
    if (par->right == loc)
    {
        par->right = NULL;
    }
}

void case2(TreeNode *loc, TreeNode *par)
{
    TreeNode *child;
    if (loc->left != NULL)
        child = loc->left;
    else if (loc->right != NULL)
        child = loc->right;

    if (par->left == loc)
        par->left = child;
    if (par->right == loc)
        par->right = child;
}

void case3(TreeNode *loc, TreeNode *par)
{
    TreeNode *succ = loc, *parSucc = par;
    while (succ->left != NULL)
    {
        parSucc = succ;
        succ = succ->left;
    }
}

```

```

    }
    if (succ->right == NULL)
        case1(succ, parSucc);
    else if (succ->right != NULL)
        case2(loc, par);

    if (par->left == loc)
        par->left = succ;
    else if (par->right == loc)
        par->right = succ;

    succ->left = loc->left;
    succ->right = loc->right;

    if (loc == root)
        root = NULL;
}

void delete (int x)
{
    TreeNode *par, *loc;
    find(x, &loc, &par);
    if (loc == NULL)
    {
        printf("Doesn't exist");
        return;
    }
    if (loc->left == NULL && loc->right == NULL)
        case1(loc, par);
    else if (loc->left == NULL && loc->right != NULL)
        case2(loc, par);
    else if (loc->left != NULL && loc->right == NULL)
        case2(loc, par);
    else if (loc->left != NULL && loc->right != NULL)

```

```

        case3(loc, par);

    free(loc);
}

int inorder(TreeNode *temp, int *count)
{
    if (temp == NULL)
    {
        return 0;
    }
    // LNR
    inorder(temp->left, count);
    (*count)++;
    inorder(temp->right, count);
}

void countLeaf(TreeNode* root, int* leafCount)
{
    if(root==NULL)
        return;

    countLeaf(root->left, leafCount);
    countLeaf(root->right, leafCount);
    if(root->left==NULL && root->right==NULL){
        (*leafCount)++;
    }
}

int height(TreeNode* root)
{
    if(root==NULL)
        return 0;

    int leftHeight = height(root->left);

```

```

    int rightHeight = height(root->right);
    int maxi=leftHeight;
    if(rightHeight > leftHeight)
        maxi=rightHeight;
    int ans = maxi+1;
    return ans;
}

int main()
{
    insert(10);
    insert(5);
    insert(50);
    insert(20);
    insert(80);
    insert(70);
    insert(100);
    insert(60);
    TreeNode *temp = root;
    int count=0,lc=0;
    Print(temp);
    inorder(temp,&count);
    printf("\n Count: %d\n", count);
    // printf("\n");

    countLeaf(temp, &lc);
    printf("Leaf Node Count: %d\n", lc);

    delete (20);
    Print(temp);

    printf("\nHeight: %d", height(temp));
    return 0;
}

```

```

5 10 20 50 60 70 80 100
Count: 8
Leaf Node Count: 4
5 10 50 60 70 80 100
Height: 5

```