1) write a c++ program for the implementation of Array

```
#include <iostream>
using namespace std;
int main() {
 int numbers[5]= {7, 5, 6, 12, 35};
cout<<" The numbers are:";
for (const int &n : numbers){
   cout<<n<<" ";
}
cout<<"\n The numbers are:";
for (int i=0; i<5; ++i){
   cout<<numbers[i]<<" ";
}
return 0;
}
```

Output

The numbers are: 7 5 6 12 35
The numbers are: 7 5 6 12 35

2) write a c++ program for the implementation of special matrices.

```
#include <iostream>
using namespace std;
int main()
{
int r, c, a[100][100], b[100][100], sum[100][100], i, j;
cout<<"Enter number of rows:";
  cin>>r;
cout<<"Enter number of columns:";
  cin>>c;
cout<<endl<<"Enter elements of 1st matrix:"<<endl;
for (i=0; i<r; ++i)
  for (j=0; j<c; ++j) {
  cout<<"Enter element a" << i+1 << j+1 <<":";
        cin>> a[i][j];
}
cout<<"endl<<"Enter elements of 2nd matrix:"<<endl;
 for (i=0; i<r; ++i)
    for (j=0; j<c; ++j)
 {
cout<<"Enter element b" << i+1 << j+1 <<":";
    cin>> b[i][j];
}
 for (i=0; i<r; ++i)
   for (j=0; j<c; ++j)
    sum[i][j] = a[i][j] + b[i][j];
```

```
cout<<endl << "sum of two matrix is :" <<endl;
 for (i=0 ; i<r ; ++i)
     for (j=0 ; j<c ; ++j)
     {
     cout<< sum [i] [j]<< " ";
     if (j==c-1)
         cout<<endl;
 }
return 0;
}
```

Output

```
Enter number of rows : 3
Enter number of columns : 2

Enter elements of 1st matrix :
Enter element a11 : 2
Enter element a12 : 4
Enter element a21 : 5
Enter element a22 : 6
Enter element a31 : 5
Enter element a32 : 8

Enter elements of 2nd matrix :
Enter element b11 : 6
Enter element b12 : 5
Enter element b21 : 4
Enter element b22 : 3
Enter element b31 : 8
Enter element b32 : 9

Sum of two matrix is :
8  9
9  9
13 17
```

3) write a C++ program for the implementation of sparse matrices.

```
#include <iostream>
using namespace std;
int main(){
 int a[10][10] = {{0,0,9}, {5,0,8}, {7,0,9}};
int i, j, count=0;
int row =3, col= 3;
for (i=0; i< row; ++i){
 for (j=0; j<col; ++j){
      if (a[i][j]==0)
        count++;
 }
 }
 cout<<"The matrix is:"<<endl;
for (i=0; i<row; ++i){
   for (j=0; j<col; ++j) {
      cout<<a[i][j]<<" ";
 }
 Cout<<endl;
 }
Cout<<"The number of zeros in the matrix are"<< Count.
                                    <<endl;
if (count > ((row * col)/2))
Cout<<" This is a sparse matrix" <<endl;
 else
cout<<" This is not a sparse matrix" <<endl;
  return 0;
}
```

Output

The matrix is :

0	0	9
5	0	8
7	0	0

The number of zeros in the matrix are 5

This is a sparse matrix

4) write a C++ program for the implementation of string

```
#include <iostream>
#include <cstring>
using namespace std;
int main() {
  char str1[10] = "Hello";
  char str2[10] = "World";
  char str3[10];
  int len;
strcpy (str3, str1);
cout<<"strcpy (str3, str1) :" << str3 <<endl;
strcat (str1, str2);
cout<<" strcat (str1, str2):" <<str1 <<endl;
len = strlen (str1);
cout<<" strlen (str1) :" << len <<endl;
return 0;
}
```

Output

strcpy (str3, str1) : Hello
strcat (str1, str2) : Hello world
strlen (str1) : 10

5) write a c++ program to implement the following using array

a) stack

```
#include <iostream>
using namespace std;
int stack[100], n=100, top=-1;
void push(int val){
  if (top>=n-1)
 cout<< " Stack overflow"<<endl;
  else {
     top++;
     stack[top]=val;
 }
}
void pop(){
      if (top<=-1)
    cout<<"Stack underflow" <<endl;
  else {
   cout<<" The popped element is" << stack[top] << endl;
   top--;
}
}
void display (){
   if (top>=0){
    cout<<" Stack elements are :";
 for (int i=top; i>=0; i++)
     cout << stack[i] <<" ";
     cout<<endl;
}
```

```
else
  cout<<"Stack is empty";
}
int main(){
  int ch, val;
cout<<"1) Push in stack"<<endl;
cout<<"2) Pop from stack"<<endl;
cout<<"3) Display stack"<<endl;
cout<<" 4) Exit" <<endl;
do {
 cout<<" Enter choice :" <<endl;
  cin>>ch;
 Switch (ch) {
  case 1 : {
     cout<<" Enter value to be pushed:"<<endl;
       cin>>val;
       Push (val);
       break;
  }
 cas 2 : {
       Pop();
       break;
 }
 case 3 : {
       display();
       break;
 }
 case 4 : {
   cout<<" Exit" <<endl;
       break;
```

```
}
default : {
        cout<< " Invalid choice " <<endl;
}
}
}
while (ch! =4) ;
  return 0;
}
```

Output

1) Push in stack
2) Pop from stack
3) Display stack
4) Exit

Enter choice :
1

Enter value to be pushed :
5

Enter choice :
1

Enter value to be pushed :
6

Enter choice :
1

Enter value to be pushed :
8

Enter choice :
1

Enter value to be pushed :
6

Enter choice :
2

The popped element is 6
Enter choice :
3

Stack elements are:
8 6 5

Enter choice :
4
Exit

(b) Queue.

```
#include <iostream>
using namespace std;
int queue[100], n=100, front=-1, rear=-1;
void Insert (){
    int val;
    if (rear==n-1)
Cout<<" Queue Overflow"<<endl;
else {
    if (front==-1)
    front=0;
  cout <<" Insert the element in queue:" <<endl;
  Cin >> val;
  rear++;
  queue[rear] = val;
}
}
void Delete () {
    if (Front == -1 || front > rear){
    cout << "Queue Underflow",
    return ;
}
else {
 Cout << "Elements deleted from queue is :" << queue[front]<<endl
 front ++;
}
}
void Display(){
    if (front == -1)
```

```
cout<<" Queue is empty" << endl;
else {
   cout <<" Queue elements are :";
 for (int i= front ; i<= rear ; i++)
    cout << queue [i]<<" ";
      cout<<endl;
}
}
int main(){
   int ch;
   cout<<"1) Insert element to queue" <<endl;
   cout <<'2) Delete element from queue"<<endl;
   cout << " 3) Display all the elements of queue"<<endl;
   cout<<" 4) Exit" << endl;
do{
   cout<<" Enter your choice: " <<endl;
     cin>>ch;
 switch (ch) {
 case 1: Insert();
  break;
 case 2: Delete();
  break;
  case 3: Display();
   break;
 case 4: cout<<" Exit" <<endl;
  break;
 default : cout<<" Invalid choice" <<endl;
} }
 while (ch!=4);
  return 0;
}
```

Output

```
1) Insert element to queue
2) Delete element from queue
3) Display all the elements of queue
4) Exit
Enter your choice:
1
Insert the element to queue:
4
Enter your choice:
1
Insert the element to queue:
6
Enter your choice:
1
Insert the element to queue:
8
Enter your choice:
1
Insert the element to queue:
6
Enter your choice:
2
Element deleted from queue is:4
Enter your choice:
3
Queue elements are: 6 8 6
Enter your choice:
4
Exit
```

6) write a C++ Program to implement the following using
a) single linked list

```
#include <iostream>
using namespace std;
class Node
{
 Public :
      int data;
      Node * next;
};
void insertNode (Node ** head, int data){
      Node* new_node = new Node();
      new_node -> data = data;
      new_node -> next = *head;
      *head = new_node;
}
void deleteNode(Node ** head){
      Node* temp = *head;
 if (*head == NULL){
  cout<<" Deletion from Empty Singly linked Not Possible";
  return;
 }
 delete(temp);
 }
void display (Node* temp) {
 Cout<<" Linked list: ";
  int count = 0;
```

```
while (temp != NULL){
cout << temp-> data << " ";
temp = temp-> next;
count ++;
}
cout << "\n There are" << count << " Items in linked list\n";
cout << endl;
}
int main(){
Node *head = NULL;
insertNode (& head, 50);
insertNode (& head, 40);
insertNode (& head, 30);
insertNode (& head, 20);
insertNode (& head, 10);
display (head);

deleteNode(& head);
deleteNode (& head);
deleteNode (& head);
display (head);
return 0;
}
```

Output

linked list: 10 20 30 40 50
There are 5 items in linked list

b) Doubly linked list

```
#include <iostream>
using namespace std;
struct Node {
    int data;
    Struct Node* next;
    Struct Node* Prev;
};
void insertfront (Struct Node** head, int data){
        Struct Node* newNode = new Node;
        newNode -> data = data;
        newNode -> next = (* head);
        newNode -> Prev = NULL;
  if ((*head) != NULL)
      (*head) -> Prev = newNode;
    (* head) = newNode;
  }
void insertAfter(Struct Node* Prev_node, int data){
    if (Prev_node == NULL){
      cout<< "Previous node cannot be null";
    return;
  }
Struct Node* newNode = new Node;
newNode -> data = data;
newNode -> next = Prev_node -> next;
Prev_node -> next = newNode;
newNode -> Prev = Prev_node;
 if (newNode -> next != NULL)
```

```
    newNode -> next -> Prev = newNode;
}
void insertEnd(struct Node** head, int data){
  struct Node* newNode = new Node;
   newNode -> data = data;
   newNode -> next = NULL;
  Struct Node* temp = *head;
 if (*head == NULL){
      newNode -> Prev = NULL;
       *head = newNode;
       return;
 }
while (temp -> next != NULL)
      temp = temp -> next;
  temp -> next = newNode;
  newNode -> Prev = temp;
}
void deleteNode (struct Node ** head, struct Node* del_node){
    if (*head == NULL || del_node == NULL)
    return;
 if (*head == del_node)
    *head = del_node -> next;
 if (del_node -> next != NULL)
    del_node -> next -> Prev = del_node -> Prev;
 if (del_node -> Prev != NULL)
    del_node -> Prev -> next = del_node -> next;
  delete del_node;
}
```

```
void displayList (struct Node* node){
  Struct Node* last;
  while (node != NULL){
      cout << node -> data << "->";
      last = node;
      node = node -> next;
}
if (node == NULL)
      cout << "NULL\n";
}
int main(){
Struct Node * head = NULL;
insertEnd (&head, 5);
insertFront (&head, 1);
insertFront (&head, 6);
insertFront End (&head, 9);
insertAfter (head, 11);
insertAfter (head -> next, 15);
displayList (head);
deleteNode (&head, head -> next -> next -> next -> next -> next);
displayList (head);
 return 0;
}
```

Output

6→11→15→1→5→9→NULL

6→11→15→1→5→ NULL

(C) Circular linked list.

```
#include <iostream>
using namespace std;
Struct Node {
    int data;
    Struct Node* next;
};
Struct Node* insertInEmpty (Struct Node* last, int new_data):
 if(last != NULL)
  return last;
 Struct Node* temp = new Node;
  temp-> data = new_data;
   last = temp;
   last -> next = last;
   return last;
}
Struct Node* insertAtBegin (Struct Node* last, int new_data){
 if (last = = NULL)
  return insertInEmpty (last, new_data);
 Struct Node* temp = new Node;
  temp-> data = new_data;
  temp-> next = last -> next;
   last -> next = temp;
   return last;
}
Struct Node* insertAtEnd (Struct Node* last, int new_data) {
 if (last = = NULL)
  return insertIn Empty (last, new_data);
 Struct Node* temp = new Node;
```

```
    temp->data = new_data;
    temp->next = last->next;
    last->next = temp;
    last = temp;
    return last;
}
Struct Node* insertAfter (struct Node* last, int new_data, int
                                                after_item) {
  if (last==NULL)
      return NULL;
  Struct Node* temp, *P;
   P= last->next;
  do {
   if (P->data== after_item){
    temp = new Node;
    temp->data = new_data;
    temp->next = P->next;
    P->next = temp;
  if (P==last)
    last = temp;
    return last;
}
  P= P->next;
}
while (P!= last->next);
cout<<"The node with data" << after_item<<" is not
            Present in the list" <<endl;
    return last;
}
```

```
void traverseList(struct Node* last){
    Struct Node* p;
if (last == NULL) {
 cout<<" Circular linked list is empty" <<endl;
    return;
}
P=last->next;
do {
 cout<< P->data <<"=>";
    P=P->next;
}
while (P!=last->next);
if (P==last->next)
  cout<< P->data;
  cout<<"\n\n";
}
void deleteNode(Node** head, int key){
if (*head == NULL)
   return;
if ((*head)->data == key && (*head)->next == *head){
   delete* head;
   *head = NULL;
}
Node* last =*head, *d;
if ((*head)->data == key){
 while (last->next != *head)
   last= last->next;
   last->next= (*head)->next;
    delete *head;
    *head= last->next;
}
```

```
while (last->next != *head && last->next->data != key){
    last = last->next;
}
if(last->next->data == key){
    d = last->next;
    last->next = d->next;
cout<<" The node with data"<<key<<" deleted from the
        list" <<endl;
delete d;
cout<<endl;
cout<<" The node with data"<<key<<" not found in the
        list"<<endl;
delete d;
cout<<endl;
cout<<"Circular linked list after deleting"<<key<<" is as
        follows:"<<endl;
    traversalist(last);
}
else {
cout<<"The node with data"<<key<<" not found in the list"
                                <<endl;

}
}
int main(){
  Struct Node* last = NULL;
last = insertInEmpty(last, 30);
last = insertAtBegin(last, 20);
last = insertAtBegin(last, 10);
last = insertAtEnd(last, 60);
```

```
last = insertAfter (last, 50, 40);
cout << "The circular linked list created is as follows"
                                << endl;
traverselist(last);
deleteNode(& last, 10);
return 0;
}
```

Output

The circular linked list created is as follows:

10 => 20 => 30 => 40 => 50 => 60 => 10

The node with data 10 deleted from the list

Circular linked list after deleting 10 is as follows:

20 => 30 => 40 => 50 => 60 => 20

7) write a C++ Program to implement stack using linked list.

```
#include <iostream>
using namespace std;
Stud struct Node
{
int data;
Node *link;
};
Node *top = NULL;
bool isempty()
{
if (top == NULL)
return true; else
return false;
}
void Push(int value)
{
Node *ptr = new Node();
ptr->data = value;
ptr->link = top;
top = ptr;
}
void pop() {
if (isempty())
cout<<"stack is Empty";
else {
Node *ptr = top;
top = top -> link;
delete (ptr);
```

```
}
}
void showTop(){
if (isempty())
cout<<"Stack is Empty";
else
  cout<<"Element at top is :" << top->data;
}
void displaystack()
{
if (isempty())
cout<<"Stack is Empty";
else
{
Node * temp = top;
while (temp != NULL) {
cout<< temp-> data << " ";
temp = temp-> link;
}
cout<<"\n";
}
}
int main()
{
int choice, flag=1, value;
while (flag==1)
{
cout<<"\n1. Push 2. Pop 3.showTop 4.displaystack 5.exit\
  cin>> choice;
```

```
switch (choice)
{
case 1 : cout<<"Enter value:\n";
        cin >> value;
        push(value);
        break;
case 2: pop();
        break;
case 3 : showTop();
        break;
case 4 : displayStack();
        break;
case 5: flag=0;
        break;
}
}
return 0;
}
```

Output

1. Push 2. Pop 3. showTop 4. displaystack 5. exit
1
Enter value :
4
1. Push 2. POP 3. showTOP 4. displaystack 5. exit
1
Enter value :
6
1. Push 2. POP 3. showTOP 4. displaystack 5. exit
1
Enter value :
2
1. Push 2. POP 3. showTOP 4. displaystack 5. exit
2
1. Push 2. POP 3. showTOP 4. displaystack 5. exit
3
Elements at top is 6.
1. push 2. Pop 3. showTOP 4. displaystack 5. exit
4
6 4
1. push 2. pop 3. showTOP 4. displaystack 5. exit
5

8) write a C++ Program to implement queue using linked list.

```
#include <iostream>
using namespace std;
Struct Node {
    int data;
    Node* link;
};
Node* front = NULL;
Node* rear = NULL;
bool isempty () {
    if (front == NULL && rear == NULL)
        return true;
else
    return false;
}
void enqueue (int value) {
    Node* ptr = new Node();
    Ptr -> data = value;
    Ptr -> link = NULL;
if (front == NULL) {
    front = ptr;
    rear = Ptr;
}
else {
    rear -> link = Ptr;
    rear = ptr;
}
}
```

```
void ShowFront (){
  if (isempty ())
   cout<<" Queue is empty \n";
else
   cout<<" Element at Front is :" << front -> data <<endl;
}
void displayQueue (){
    if (isempty())
  cout <<" Queue is empty \n";
else {
    Node* ptr = front;
   while (ptr != NULL){
      cout << ptr-> data <<" ";
        ptr = ptr-> link;
}
 cout<<endl;
}
}
int main(){
int choice, flag=1, value;
while (flag == 1){
Cout<<"\n 1. Enqueue 2. Dequeue 3. show Front
              4. Display Queue 5. Exit\n";
 Cin>>choice;
Switch (choice){
Case 1 :
    Cout << " Enter value :";
     Cin >> value;
   enqueue (value);
    break;
```

```
case 2 :
        dequeue();
        break;
case 3:
        showfront();
        break;
case 4:
        displayQueue();
        break;
case 5:
        flag = 0;
        break;
default :
        cout << "Invalid choice\n";
        break;
}
}
return 0;
}
```

Output

1. Enqueue 2. Dequeue 3. Show Front 4. Display Queue 5. Exit

1

Enter value : 4

1. Enqueue 2. Dequeue 3. Show Front 4. Display queue 5. Exit

1

Enter value : 6

1. Enqueue 2. Dequeue 3. show Front 4. Display Queue 5. Exit

1

Enter value : 8

1. Enqueue 2. Dequeue 3. Show Front 4. Display Queue 5. Exit

2

1. Enqueue 2. Dequeue 3. show Front 4. Display Queue 5. Exit

3

Element at Front is : 6.

1. Enqueue 2. Dequeue 3. Show Front 4. Display Queue 5. Exit

4

6 8

1. Enqueue 2. Dequeue 3. Show Front 4. Display Queue 5. Exit

5

a) write a C++ program to implement binary tree.

```
#include <stdlib.h>
#include <iostream>
using namespace std;
Struct node {
  int data;
 Struct node * left;
 Struct node *right;
};
Struct node *newNode (int data) {
Struct node *node = (struct node*) malloc (sizeof (struct
                                                node));
node -> data = data;
node -> left = NULL;
node -> right = NULL;
return (node);
}
Void traverse PreOrder (struct node *temp){
  if (temp != NULL) {
    Cout<<" " << temp->data;
    traverse PreOrder (temp -> left);
    traverse PreOrder (temp -> right);
}
}
void traverse InOrder (struct node *temp){
 if (temp != NULL) {
   traverseInOrder (temp -> left);
  cout << " " << temp -> data;
  traverseIn Order (temp -> right);
}
}
```

```
void traversePostOrder (struct node *temp){
  if (temp != NULL){
    traverse PostOrder (temp->left);
    traverse PostOrder (temp->right);
    cout<<" " << temp->data;
  }
}

int main(){
  Struct node *root = newNode( );
  root -> left = newNode (2);
  root -> right = newNode (3);
  root -> left -> left = newNode(4);
  cout<<" Preorder traversal :";
  traversePreOrder (root);
  cout<<"In Inorder traversal :";
  traverseInorder (root);
  Cout<<"In Postorder traversal: ";
  travese PostOrder (root);
}
```

Output

Preorder traversal : 1 2 4 3
In order traversal : 4 2 1 3
Postorder traversal : 4 2 3 1

10) write C++ Program for implementing the following Sorting methods

a) Bubble sort.

```
#include <iostream>
using namespace std;
int main() {
int n, i, arr[50], j, temp;
Cout<<" Enter the size : ";
cin>>n;
cout<< "Enter "<<n<<" Numbers : ";
for (i=0 ; i<n; i++)
      cin>>arr[i];
Cout<<"\n Sorting the Array using Bubble sort \n";
for (i=0; i<(n-1); i++) {
  for (j=0; i<(n-i-1); j++)
  {
    if (arr[j]> arr[j+1]) {
    temp = arr[j];
    arr[j] = arr[j+1];
    arr[j+1] = temp;
}
}
}
Cout<<"\n Array Sorted Successfully! \n";
Cout<<"\n The New Array is : \n";
for (i=0; i<n; i++)
      cout<<arr[i]<<" ";
Cout<<endl;
return 0;
}
```

Output:

Enter the size : 8
Enter 8 Numbers : 56 7 8 34 23 12 24 46

Sorting the Array using Bubble Sort

Array Sorted Successfully!
The New Array is:
7 8 12 23 24 34 46 56

b) selection sort.

```
#include <iostream>
using namespace std;
int main()
{
int tot, arr[50], i, j, temp, small, chk, index;
cout<<"Enter the Size of Array :";
cin>>tot;
cout<<"Enter"<<tot<<"Array Elements :";
for (i=0; i<tot; i++)
 cin>>arr[i];
for (i=0; i<(tot-1); i++){
 chk=0;
 small= arr[i];
 for (j=(i+1); j<tot; j++){
  if(small>arr[j]){
   small= arr[j];
   chk++;
   index=j;
  }
 }
 if(chk!=0){
   temp = arr[i];
   arr[i] = small;
   arr[index] = temp;
 }
}
```

```
cout<<"\n Sorted Array is:\n";
for (i=0; i<tot; i++)
    cout<<arr[i]<<" ";
cout<<endl;
return 0;
}
```

Output

Enter the size of Array : 6
Enter 6 Array Elements : 12 34 25 7 8 23

Sorted Array is :
7 8 12 23 25 34

(c) Insertion Sort.

```
#include <iostream>
using namespace std;
int main()
{
int arr[50], tot, i, j, k, elem, index;
cout<<" Enter the size for Array:";
cin>>tot;
cout<<"Enter"<<tot<<" Array Elements:";
for(i=0; i<tot; i++)
  Cin>>arr[i];
for(i=1; i<tot; i++){
 elem= arr[i];
 if (elem<arr[i-1]){
 for (j=0; j<=i; j++)
 {
  if (elem <arr[j])
  {
   index = j;
   for (k=i; k>>j; k--)
       arr[k]= arr[k-1];
       break;
  }
  }
  }
 else
   Continue;
 arr [index] = elem;
}
```

```
cout<<"\n The New Array is :\n";
  for(i=0; i<tot; i++)
      cout<<arr[i]<<" ";
  cout<<endl;
  return 0;
}
```

Output

Enter the size for Array : 9
Enter 9 Array Elements: 12, 34 24 8 9 46 15
34 45

The new Array is :
8 9 12 15 24 34 34 45 46.

(d) Quick Sort

```
#include <iostream>
using namespace std;
void swap (int *a, int *b){
    int t = *a;
    *a = *b;
    *b = t;
}
void PrintArray(int array[], int size){
 int i;
 for (i=0; i<size; i++)
    cout<< array [i] << " ";
    cout<<endl;
}
int Partition (int array[], int low, int high){
 int pivot = array[high];
 int i= (low-1);
 for (int j= low; j<high; j++){
  if (array [j]<=Pivot) {
    i++;
   swap(&array[i], &array[j]);
  }
 }
 swap(&array [i+1], &array [high]);
 return (i+1);
}
void quicksort(int array [], int low, int high){
  if(low<high) {
```

```
    int pi= partition (array, low, high);
    quicksort (array, low, pi-1);
    quicksort (array, pi+1, high);
}
}
int main(){
 int data[] = {8,7,6,10,9,2};
 int n= sizeof(data)/sizeof(data[0]);
Cout<< "Unsorted Array :\n";
 PrintArray (data,n);
 quickSort(data, 0, n-1);
Cout<<"Sorted array in ascending Order :\n";
  PrintArray (data, n);
}
```

Output

Unsorted Array:

8 7 6 1 0 9 2

Sorted array in ascending order:

0 1 2 6 7 8 9

(e) Shell sort

```
#include <iostream>
using namespace std;
void shellSort (int array[], int n){
for (int interval= n/2; interval>0; interval/=2)
{
 for(int i= interval; i<n; i+=1){
  int temp= array[i];
  int j;
 for (j=i; j>= interval && array[j- interval]> temp;
                                    j-= interval){

     array[j]= array[j- interval];
 }
 array[j]= temp;
}
}
}
void PrintArray (int array[], int size){
 int i;
 for (i=0; i<size; i++)
    cout<< array[i] << " ";
    cout<<endl;
 }
 int main(){
 int data[]= {9, 8, 3, 7, 5, 6, 4, 1};
 int size = sizeof (data) /sizeof (data[0]);
 ShellSort (data, size);
 Cout<< "Sorted array:\n";
  PrintArray (data, size);
}
```

Output

Sorted Array :

1 3 4 5 6 7 8 9

f) merge sort

```
#include <iostream>
using namespace std;
void merge (int arr[], int p, int q, int r) {
  int n1 = q-p+1;
  int n2 = r-q;
 int L[n1], m[n2];
for(int i=0; i<n1; i++)
  L[i] = arr [p+i];
for (int j=0; i<n2; j++)
  m[j] = arr [q+1+j];
  int i, j, k;
  i = 0;
  j = 0;
  k = p;
 while (i<n1 && j<n2) {
  if (L[i]<=m[j]) {
    arr[k] = L[i];
    i++;
}
 else {
  arr[k] = m[j];
    j++;
}
k++;
}
while (i<n1) {
  arr[k] = L[i];
   i++;
   k++;
}
```

```
}
void mergesort(int arr[], int l, int r){
 if(l<r){
  int m=l+(r-l)/2;
   mergesort(arr, l, m);
   mergesort(arr, m+1, r);
   merge(arr, l, m, r);
 }
}
Void PrintArray(int arr[], int size){
 for(int i=0; i< size; i++)
  cout<<arr[i]<<" ";
  cout<<endl;
}
int main(){
 int arr[]= {6, 5, 12, 10, 9, 1};
 int size = sizeof(arr)/sizeof(arr[0]);
 mergesort(arr, 0, size-1);
  cout<<"Sorted array:\n";
  PrintArray(arr, size);
 return 0;
}
```

Output

Sorted array :

1 5 6 9 10 12

(9) Heap Sort

```
#include <iostream>
using namespace std;
void heapify (int arr[], int n, int i) {
  int largest = i;
  int left = 2* i+1;
  int right = 2* i+2;
if (left<n && arr[left] > arr[largest])
    largest = left;
if (right<n && arr[right] > arr[largest])
    largest = right;
 if (largest != i) {
    swap(arr[i], arr[largest]);
 heapify (arr, n, largest);
}
}
void heapSort (int arr[], int n){
 for (int i=n/2-1 ; i>=0; i--)
   heapify (arr, n, i);
 for (int i=n-1; i>0 ;i--){
   swap(arr[0], arr[i]);
   heapify (arr, i, 0);
 }
}
void printArray (int arr[], int n) {
 for (int i=0 ; i<n ;i++)
    cout<< arr[i] << " ";
  cout<<endl;
}
```

```
int main() {
    int arr[] = {12, 11, 13, 5, 6, 7};
    int n= sizeof (arr) / sizeof (arr[0]);
    cout<< " Original array: ";
    PrintArray (arr, n);
    heapSort (arr, n);
    Cout<< " Sorted array : ";
    PrintArray (arr, n);
    return 0;
}
```

Output

Original array : 12 11 13 5 6 7

Sorted array : 5 6 7 11 12 13

11) Programs on Linear Search and Binary search using recursion and iteration.

Linear Search using recursion.

```
#include <iostream>
using namespace std;
int LinearSearch (int arr[], int n, int key)
{
 for(int i=0; i<n; i++)
  {
   if(key == arr[i])
       return i;
  }
  return -1;
}
int main() {
int array[] = {50,90, 30, 70, 60};
int key=0;
cout<< "Enter Search Element :";
 cin>> key;
int size= sizeof(array) /sizeof(array[0]);
int index= linearSearch (array, size, key);
if (index==-1)
  cout<< key<< "Not found"<<endl;
else
  cout<< key<< "Found at Index =" << index << endl;
  return 0;
}
```

Output

Enter Search Element : 30
30 Found at Index = 2

Linear Search using iteration

```
#include <iostream>
using namespace std;
int main()
{
int i, n, key, position = -1;
cout<<" Enter the number of elements in the array"<<endl;
  cin>>n;
int array[n];
cout<<" Enter the elements in the array"<<endl;
for (i=0; i<n; i++) {
   cin>>array[i];
}
cout<<"Enter the element to be searched"<<endl;
   cin>>key;
for (i=0; i<n; i++) {
  if (key== array[i]) {
   Position = i;
    break;
}
}
if (Position == -1) {
cout<<'key is not found in the array"<<endl;
}
else {
 cout<<" key found at Position"<< Position <<endl;
}
return 0;
}
```

Output

Enter the number of elements in the array
6
Enter the elements in the array
34 23 6 15 25 18
Enter the element to be searched
23
key found at Position 1

Binary Search using iteration.

```
#include <iostream>
using namespace std;
int main()
{
int i, arr[10], num, first, last, middle;
cout<< "Enter 10 elements (in ascending order):";
 for (i=0; i<10; i++)
      cin>>arr[i];
cout<<" \n Enter Element to be Search:";
  cin>>num;
 first =0;
  last = 9;
middle = (first+ last)/2;
while (first <= last)
 {
 if (arr[middle] <num)
      first = middle +1;
 else if (arr[middle] == num)
{
Cout<<"\n The number, "<<num<<" found at position"
                                    <<middle +1;

 break;
 }
 else
    last = middle -1;
   middle = (first+ last)/2;
}
```

```
if (first > last)
    cout<<"In The number, "<<number" is not found
                                in given Array";

cout<<endl;
return 0;
}
```

Pinnacle Degree College
Affiliated to Osmania University
Moosapet, Hyd-18

Output.

Enter 10 elements (in ascending order:

4 6 12 15 20 30 40 45 50 55

Enter element to be Search : 30

The number, 30 found at position 6.