//1. Array Insertion and deletion using menu bar.cpp

```
#include<bits/stdc++.h>
using namespace std;
#include "windows.h"
vector<int>vec;
void display(vector<int>vec){
   if(vec.size() == 0)     cout << "Array is empty!\n" << endl;
   else{
      cout << "Array is : ";
      for(auto d:vec) cout << d << " ";
      cout << endl;
   }
}
int main(){
   while(1){
      system("CLS");
      display(vec);
      cout << "Enter I for insert elemet in an array!!\n";
      cout << "Enter D for delete element in an array!!\n";
      cout << "Enter any for Exit!!\n";
      cout << "\nEnter your choice: ";
      char ch;   cin >> ch;
```

```
        if(ch == 'i' || ch == 'I'){
            cout << "Enter any integer for insert: ";
            int x;  cin >> x;
            vec.push_back(x);
        }
        else if(ch == 'd' || ch == 'D'){
            if(vec.size() == 0){
                cout << "Array is already empty!!\n\n";
                Sleep(1000);
            }
            else vec.erase(vec.begin()+vec.size()-1);
        }
        else break;
    }
}
```

//2.Sum of boundary elements in a matrix using random number.cpp

```
#include<bits/stdc++.h>
using namespace std;
int main(){
   cout << "Enter row number of a matrix: ";
   int row;   cin >> row;
   cout << "Enter coloum number of a matrix: ";
   int col;   cin >> col;
   int matrix[row][col];
   int sum = 0;
   for(int i = 0; i < row; i++){
      for(int j = 0; j < col; j++){
         matrix[i][j] = rand() % 100;
         cout << matrix[i][j] << " ";
         if(j == 0) sum += matrix[i][j];
         else if(j == col-1) sum += matrix[i][j];
         else if(i == 0 && i < col-1) sum += matrix[i][j];
         else if(i == row-1 && j < col-1) sum += matrix[i][j];
      }cout << endl;
   }
   cout << "Sum of boundary elements : " << sum << endl;
```

//3. Pattern matching with string and replace substring.cpp

```
#include <bits/stdc++.h>
using namespace std;
int main(){
   cout << "Input Main String: ";
   string main;    cin >> main;
   cout << "Input Sub-string: ";
   string sub; cin >> sub;
   vector<int>vec(sub.size(),0);
   int i = 0, j = 1;
   while(j < sub.size()){
      if(sub[i] == sub[j]){
         i++;
         vec[j] = i;
         j++;
      }
      else if(i == 0) j++;
      else i = vec[i-1];
   }

   i = j = 0;
```

```
    int cunt = 0;
    while(j<main.size()){
        if(sub[i] == main[j]){
            i++;
            j++;
        }
        else if(i == 0) j++;
        else i = vec[i-1];
        if(i == sub.size()){
            for(int p = j-sub.size(); p < j; p++){
                main[p] = '*';
            }
            i = vec[i-1];
        }
    }
    cout << "Main String: " << main ;
}
```

//4.Bubble Sort.cpp

```
#include<bits/stdc++.h>
using namespace std;
int main(){
   cout << "Enter array elements number: ";
   int n;  cin >> n;
   int array[n];
   for(int i = 0; i < n; i++){
      cin >> array[i];
   }
   for(int i = 0; i < n-1; i++){
      for(int j = 0; j < n-1; j++){
         if(array[j] > array[j+1])
            swap(array[j], array[j+1]);
      }
   }
   for(int i = 0; i < n; i++){
      cout << array[i] << " ";
   }
}
```

//5.Insertion Sort.cpp

```
#include<bits/stdc++.h>
using namespace std;
int main(){
   cout << "Enter number of array elements: ";
   int n;  cin >> n;
   int array[n];
   for(int i=0; i<n; i++)  cin >> array[i];
   int key,j;
   for(int i = 1; i<n; i++){
      key = array[i];
      j = i-1;
      while(j>=0 && array[j] > key){
         array[j+1] = array[j];
         j = j-1;
      }
      array[j+1] = key;
   }
   cout << "\nSorted Array is: ";
   for(auto d:array)     cout << d << " ";
}
```

//6.Selection Sort.cpp

```
#include<bits/stdc++.h>
using namespace std;
int main(){
   cout << "Enter number of array element: ";
   int n;  cin >> n;
   int array[n];
   for(auto &d:array)   cin >> d;
   int index = -1;
   for(int i = 0; i < n; i++){
      int min = array[i];
      for(int j = i+1; j<n; j++){
         if(array[j] < min){
            min = array[j];
            index = j;
         }
      }
      swap(array[i],array[index]);
   }
   for(auto d:array)   cout << d << " ";
}
```

//7.Binary Search.cpp

```
#include<bits/stdc++.h>
using namespace std;
int main(){
   cout << "Enter number of array element: ";
   int n;  cin >> n;
   int array[n];
   for(auto &d:array)  cin >> d;
   sort(array+0,array+n);
   cout << "Which element you want to Search: ";
   int element;    cin >> element;
   int first = 0, last = n-1;
   while(first <= last){
      int mid = (first + last) / 2;
      if(element > array[mid])    first = mid+1;
      else if(element < array[mid]) last = mid-1;
      if(element == array[mid]){
         cout << element <<" is found in index: " << mid+1 << endl;
         break;
      }
   }
}
```

//8.Linear Search.cpp

```
#include<bits/stdc++.h>
using namespace std;
int main(){
   cout << "Enter number of elements: ";
   int n;  cin >> n;
   int array[n];
   for(auto &d:array)  cin >> d;
   cout << "Which element you want to search: ";
   int element;  cin >> element;
   for(int i = 0; i<n; i++){
      if(array[i] == element){
         cout << element <<" is found in index: " << i+1 << endl;
         break;
      }
   }
}
```

//9.Student Result.cpp

```
#include <bits/stdc++.h>
using namespace std;
int main(){
   cout << "Enter number of student: ";
   int n, temp;    cin >> n;
   temp = n;
   string name[n], grade[n];
   int roll[n], mark[n], copy[n];
   for (int i = 0; i < n; i++){
      cout << "Name of student " << i + 1 << ": ";
      cin >> name[i];

      cout << "Roll of student " << i + 1 << ": ";
      cin >> roll[i];

      cout << "Mark of student " << i + 1 << ": ";
      cin >> mark[i];
      copy[i] = mark[i];
      cout << endl;

      // For Grade
```

```
      if (mark[i] >= 80) grade[i] = "A+";
      else if (mark[i] >= 75) grade[i] = "A";
      else if (mark[i] >= 70) grade[i] = "A-";
      else if (mark[i] >= 65)  grade[i] = "B+";
      else if (mark[i] >= 60) grade[i] = "B";
      else if (mark[i] >= 55)  grade[i] = "B-";
      else if (mark[i] >= 50) grade[i] = "C+";
      else if (mark[i] >= 45) grade[i] = "C";
      else if (mark[i] >= 40) grade[i] = "D";
      else grade[i] = "F";
   }
   cout << "Roll      " << "Name    " << "Mark    " << "Grade" << endl;
   for (int i = 0; i < n; i++){
      cout << roll[i] << "       " << name[i] << "    " << mark[i] << "       " <<
grade[i] << endl;
   }
   cout << endl;
   cout << "After Sorting according to mark!!" << endl;
   sort(copy + 0, copy + n);
   reverse(copy + 0, copy + n);
   while (temp != 0){
      for (int i = 0; i < n; i++){
```

```
        for (int j = 0; j < n; j++){
            if (copy[i] == mark[j]){
                cout << roll[j] << "       " << name[j] << "    " << mark[j] <<
"      " << grade[j] << endl;
                temp--;
            }
        }
    }
  }
}
```

//10. Stack implementation using array.cpp

```
#include<bits/stdc++.h>
using namespace std;
#include "windows.h"
using ll = long long;
int m = 200;

void display(ll stack[],ll top){
   if(top==0) cout << "Stack is empty!!\n";
   else{
      cout << "Stack: ";
      for(int i = top-1; i>=0; i--) cout << stack[i] << " ";
      cout << endl;
   }
}
int main(){
   ll stack[m], top = 0;
   while(1){
      system("CLS");
      display(stack,top);
      cout << "Enter I for insert!!\n";
      cout << "Enter D for delete!!\n";
```

```
cout << "Enter any for exit!!\n";
cout << "Enter choice: ";
char ch; cin >> ch;

if(ch == 'i' || ch == 'I'){
   if(top >= m){
      cout << "Stack Overflow!!\n";
      Sleep(1000);
   }
   else{
      cout << "\nEnter element for insert: ";
      ll n; cin >> n;
      stack[top] = n;
      top++;
   }
}
else if(ch == 'd' || ch == 'D'){
   if(top == 0){
      cout << "Stack Underflow!!\n";
      Sleep(1000);
   }
   else{
```

```
            top = top - 1;
          }
        }
        else break;
    }
}
```

// 11. Queue implementation using array.cpp

```
#include <bits/stdc++.h>
using namespace std;
int m = 200;
void display(int queue[], int front, int rear){
   if (front == rear) cout << "Queue is empty!!" << endl;
   else{
      cout << "Queue: ";
      for (int i = front; i < rear; i++)
         cout << queue[i] << " ";
      cout << endl;
   }
}
int main(){
   int queue[m], rear = 0, front = 0;
   while (1){
      system("cls");
      display(queue, front, rear);
      cout << "Enter I for insert!!\n";
      cout << "Enter D for delete!!\n";
      cout << "Enter any for delete!!\n";
      cout << "Enter your choice: ";
```

```
    char ch; cin >> ch;
    if (ch == 'i' || ch == 'I'){
       if (rear == m){
          cout << "Queue Overflow!!\n";
          _sleep(1000);
       }
       else{
          cout << "\nEnter element for insert: ";
          int n; cin >> n;
          queue[rear] = n;
          rear++;
       }
    }
    else if (ch == 'd' || ch == 'D'){
       if (rear == front){
          cout << "Queue Underflow!!\n";
          _sleep(1000);
       }
       else front = front + 1;
    }
    else   break;
  }
```

//12. Circular queue implementation using array.cpp

```
#include<bits/stdc++.h>
using namespace std;
int m = 200, total = 0;
void display(int circular[],int front,int total){
   if(total==0)    cout<<"Circular queue is empty"<<endl;
   else{
      cout << "Circular queue: ";
      for(int i=0;i<total;i++)
         cout << circular[(front+i)%m] << " ";
      cout << endl;
   }
}
int main(){
   int circular[m], front = 0, rear = 0;
   while(1){
      system("cls");
      display(circular, front,total);
      cout << "Enter I for insert!!\n";
      cout << "Enter D for delete!!\n";
      cout << "Enter any for exit!!\n";
      cout << "Enter your choice: ";
```

```
char ch; cin >> ch;
if(ch == 'i' || ch =='I'){
   if(total >= m){
      cout << "Queue Overflow!!\n";
      _sleep(1000);
   }
   else{
      cout << "\nEnter element for insert: ";
      int n;  cin >> n;
      circular[rear%m] = n;
      rear++;
      total++;
   }
}
else if(ch == 'd' || ch =='D'){
   if(total < 0){
      cout << "Queue Overflow!!\n";
      _sleep(1000);
   }
   else{
      front++;
      total--;
```

```
            }
        }
        else break;
    }
}
```

//13. Convert postfix to infix using stack.cpp

```
#include<bits/stdc++.h>
using namespace std;
bool isOperand(char ch){
   return((ch >= 'a' && ch <= 'z')||(ch >= 'A' && ch <= 'Z'));
}
int main(){
   cout << "Enter postfix expression: ";
   string str; cin >> str;
   stack<string> infix;
   for(int i = 0; str[i] != '\0'; i++){
      if(isOperand(str[i])){
         string op(1,str[i]);
         infix.push(op);
      }
      else{
         string op1 = infix.top();
         infix.pop();
         string op2 = infix.top();
         infix.pop();
         infix.push("("+op2+str[i]+op1 +")");
      }
```

```
    }
    cout << "Infix expression: ";
    while(!infix.empty()){
        cout << infix.top() << endl;
        infix.pop();
    }
    cout << endl;
}
```

//14. Evaluation of postfix expresion using stack.cpp

```
#include<bits/stdc++.h>
using namespace std;
bool isOperand(char c){
   return (c >= '0' && c <= '9');
}
int main(){
   cout << "Enter postfix expresion: ";
   char ch[200];
   gets(ch);
   stack<int> infix;
   for(int i = 0; ch[i] != '\0'; i++){
      if(isOperand(ch[i])){
         int p = ch[i]-'0';
         infix.push(p);
      }
      else{
         int n = infix.top();
         infix.pop();
         int m = infix.top();
         infix.pop();
         int ans = 0;
```

```
            if(ch[i] == '+') ans = n+m;
            if(ch[i] == '-') ans = n-m;
            if(ch[i] == '*') ans = n*m;
            if(ch[i] == '/') ans = n/m;
            infix.push(ans);
        }
    }
    cout << "Postfix valu: "<< infix.top()<<endl;
}
```

//15. Link List(insert,delete,exit).cpp

```
#include<bits/stdc++.h>
using namespace std;
int cunt = 0;
struct Node{
   int data;
   Node *link;
};
//Insert First.....
struct Node *insertFirst(Node* head, int data){
   Node *newNode = new Node;
   newNode->data = data;
   cunt++;
   newNode->link = head->link;
   head->link = newNode;
   return head;
}
//Insert last.....
struct Node *insertLast(Node* head, int data){
   Node *newNode = new Node;
   Node *ptr = head;
   while(ptr->link != NULL){
```

```
        ptr = ptr->link;
    }
    newNode->data = data;
    cunt++;
    ptr->link = newNode;
    newNode->link = NULL;
    return head;
}
//Insert custom....
struct Node *insertCustom(Node *head,int data, int index){
    Node *newNode = new Node;
    Node *ptr = head;
    for(int i = 0; i < index-1; i++){
        ptr = ptr->link;
    }
    newNode->data = data;
    cunt++;
    newNode->link = ptr->link;
    ptr->link = newNode;
    return head;
}
```

```
//Delete first.....
struct Node *deleteFirst(Node *head){
   head->link = head->link->link;
   if (cunt <= 0) cunt = 0;
   else cunt--;
   return head;
}
//Delete last.....
struct Node *deleteLast(Node *head){
   Node *ptr = head;
   while(ptr->link->link != NULL){
      ptr = ptr->link;
   }
   ptr->link = NULL;
   if (cunt <= 0) cunt = 0;
   else cunt--;
   return head;
}
//Delete custom.....
struct Node *deleteCustom(Node *head,int index){
   Node *ptr = head;
   for(int i=0; i<index-1; i++){
```

```
        ptr = ptr->link;
    }
    ptr->link = ptr->link->link;
    if (cunt <= 0) cunt = 0;
    else cunt--;
    return head;
}

void display(Node *head){
    if(cunt <= 0){
        cout << "Link List is empty!" << endl;
    }
    else{
        cout << "Link list: ";
        Node *ptr = head->link;
        while(ptr !=  NULL){
            cout << ptr->data << " ";
            ptr = ptr->link;
        }
        cout << endl;
    }
}
```

```
int main(){
   Node *head = new Node;
   head->link = NULL;
   while(1){
      system("cls");
      display(head);
      cout << "\nEnter I for insert!!\n";
      cout << "Enter D for delete!!\n";
      cout << "Enter any for exit!!\n";
      cout << "Enter your choice: ";
      char ch;    cin >> ch;
      if(ch == 'i' || ch == 'I'){
         if(cunt == 0){
            cout << "Enter element for insert: ";
            int element;    cin >> element;
            insertFirst(head,element);
            cunt++;
         }
         else{
            system("cls");
            display(head);
            cout << "\nEnter F for insert first!\n";
```

```
cout << "Enter L for insert last!\n";
cout << "Enter N for insert n'th position!\n";
cout << "Enter your choice: ";
char c; cin >> c;
if(c == 'f' || c == 'F'){
   cout << "Enter element for insert first: ";
   int element;    cin >> element;
   insertFirst(head,element);
}
else if(c == 'l' || c == 'L'){
   cout << "Enter element for insert last: ";
   int element;    cin >> element;
   insertLast(head,element);
}
else if(c == 'n' || c == 'N'){
   cout << "Enter index number: ";
   int idx; cin >> idx;
   cout << "Enter element for insert " << idx <<" position: ";
   int element;    cin >> element;
   insertCustom(head,element,idx);
}
else    cout << "Invalid input Try again!\n";
```

```
    }
  }
  else if(ch == 'd' || ch == 'D'){
    system("cls");
    display(head);
    if(cunt <= 0){
      cout << "SORRY!! there is no element for delete!\n";
      cunt = 0;
      _sleep(1200);
      continue;
    }
    cout << "\nEnter F for delete first!!\n";
    cout << "Enter L for delete last!!\n";
    cout << "Enter N for delete n'th position!!\n";
    cout << "Enter your choice: ";
    char ch;    cin >> ch;
    if(ch == 'f' || ch == 'F'){
      deleteFirst(head);
    }
    else if(ch == 'l' || ch == 'L'){
      deleteLast(head);
    }
```

```
        else if(ch == 'n' || ch == 'N'){
            cout << "Enter index for delete: ";
            int idx;    cin >> idx;
            deleteCustom(head, idx);
        }
        else cout << "Invalid input Try again!\n";
    }
    else break;
  }
}
```

//16. Binary tree(insert,delete,traversal).cpp

```
#include<bits/stdc++.h>
using namespace std;
struct Node{
   int data;
   Node* left;
   Node* right;
};
struct Node *createNode(int data){
   Node *newNode = new Node;
   newNode->data = data;
   newNode->left = NULL;
   newNode->right = NULL;
   return newNode;
}
//Insert node.....
struct Node *insertNode(Node *ptr, int data){
   if(ptr == NULL) ptr = createNode(data);
   else if(ptr->data >= data) ptr->left = insertNode(ptr->left, data);
   else ptr->right = insertNode(ptr->right, data);
   return ptr;
}
```

```
//Pre-Order traversal......
void preOrder(Node *ptr){
   if(ptr != NULL){
      cout << ptr->data << " ";
      preOrder(ptr->left);
      preOrder(ptr->right);
   }
}
//Post-Order traversal......
void postOrder(Node *ptr){
   if(ptr != NULL){
      postOrder(ptr->left);
      postOrder(ptr->right);
      cout << ptr->data << " ";
   }
}
//In-Order traversal......
void inOrder(Node *ptr){
   if(ptr != NULL){
      inOrder(ptr->left);
      cout << ptr->data << " ";
      inOrder(ptr->right);
```

```
    }
}

void display(Node *root){
    cout << "Current list!!\n";
    cout << "Pre Order: ";
    preOrder(root);
    cout << endl;

    cout << "In Order: ";
    inOrder(root);
    cout << endl;

    cout << "Post Order: ";
    postOrder(root);
    cout << endl;
}
int main(){
    Node *root = NULL;
    while(1){
        system("cls");
        display(root);
```

```
        cout << "\nEnter I for insert element!\n";
        cout << "Enter any for exit!\n";
        cout << "Enter your choice: ";
        char ch;    cin >> ch;
        if(ch == 'i' || ch == 'I'){
           system("cls");
           display(root);
           cout << "Enter element for insert: ";
           int element;    cin >> element;
           root = insertNode(root, element);
        }
        else break;
    }
}
```

//19. Marge Sort.cpp

```
#include<bits/stdc++.h>
using namespace std;
void marge(int *arr, int start, int end){
   int mid = (start + end)/2;
   int len1 = mid - start +1;
   int len2 = end - mid;
   int *arr1 = new int[len1];
   int *arr2 = new int[len2];
   int index = start;
   for(int i = 0; i < len1; i++){
      arr1[i] = arr[index];
      index++;
   }
   index = mid+1;
   for(int i = 0; i < len2; i++){
      arr2[i] = arr[index];
      index++;
   }
   index = start;
   int idx1 = 0, idx2 = 0;
   while(idx1 < len1 && idx2 < len2){
```

```
        if(arr2[idx2]>arr1[idx1]){
            arr[index++] = arr1[idx1++];
        }
        else{
            arr[index++] = arr2[idx2++];
        }
    }

    while(idx1 < len1){
        arr[index++] = arr1[idx1++];
    }
    while(idx2 < len2){
        arr[index++] = arr2[idx2++];
    }
}
void margeSort(int *arr, int f, int l){
    if(f >= l) return;
    int mid = (f+l)/2;
    margeSort(arr,f,mid);
    margeSort(arr,mid+1,l);
    marge(arr,f,l);
}
```

```
int main(){
   cout << "Enter array size: ";
   int n;  cin >> n;
   int arr[n];
   for(int i = 0; i < n;i++) cin >> arr[i];

   cout << "Before sorting array!!\n";
   for(int i = 0; i < n; i++) cout << arr[i]<< " ";
   cout << endl;

   margeSort(arr,0,n-1);

   cout << "After sorting array!!\n";
   for(int i = 0; i < n; i++) cout << arr[i]<< " ";
   cout << endl;
}
```

//20. Quick Sort.cpp

```
#include<bits/stdc++.h>
using namespace std;

int partition(int arr[], int start, int end){
   int pivot = arr[start];
   int cunt = 0;
   for(int i = start+1; i<=end; i++){
      if(pivot >= arr[i]) cunt++;
   }
   int pivot_index = start + cunt;
   swap(arr[pivot_index],arr[start]);

   int i = start, j = end;
   while(i < pivot_index && j > pivot_index){
      while(arr[i] < pivot) i++;
      while(arr[j] > pivot) j--;
      if(i < pivot_index && j > pivot_index){
         swap(arr[i++],arr[j--]);
      }
   }
   return pivot_index;
```

```
}
void quickSort(int arr[], int f, int l){
   if(f>=l) return;
   int p = partition(arr,f,l);
   quickSort(arr,f,p-1);
   quickSort(arr,p+1,l);
}
int main(){
   cout << "Enter array size: ";
   int n;  cin >> n;
   int arr[n];
   for(int i=0; i<n; i++)  cin >> arr[i];

   cout << "Before quick sort!!\n";
   for(int i=0; i<n; i++) cout <<arr[i] <<" ";

   quickSort(arr,0,n-1);

   cout << "\nAfter quick sort!!\n";
   for(int i=0; i<n; i++) cout <<arr[i] <<" ";
}
```

// 22. Garph Traversal (Breadth First Search-BFS).cpp

```
#include <bits/stdc++.h>
using namespace std;
template <typename tp>
class graph{
   map<tp, list<tp>> mp;

public:
   void addEdge(int x, int y){
      mp[x].push_back(y);
      mp[y].push_back(x);
   }
   void BFS(tp src){
      map<tp, bool> visited;
      queue<tp> Q;
      Q.push(src);
      visited[src] = true;

      while (!Q.empty()){
         tp node = Q.front();
         Q.pop();
         cout << node << " ";
```

```
            for (int d : mp[node]){
                if (!visited[d]){
                    Q.push(d);
                    visited[d] = true;
                }
            }
        }
    }
};
int main(){
    graph<int> g;
    g.addEdge(0, 1);
    g.addEdge(0, 2);
    g.addEdge(1, 2);
    g.addEdge(2, 0);
    g.addEdge(2, 3);
    g.addEdge(3, 3);

    g.BFS(2);
}
```

//21. Graph representation on memory(Adjacency Matrix).cpp

```
#include<bits/stdc++.h>
using namespace std;
int graph[1000][1000];
int main(){
   int vertex,edge;
   cout << "Enter number of vertices: ";
   cin >> vertex;
   cout << "Enter number of edges: ";
   cin >> edge;
   cout<<"Enter the edges!!\n";
   int v1, v2;
   for(int i = 0; i<edge; i++){
      cin >> v1 >> v2;
      graph[v1][v2] = 1;
      graph[v2][v1] = 1;
   }
   cout << "\nAdjacency Matrix!!\n";
   for(int i = 1; i <= vertex; i++){
      for(int j = 1; j <= vertex; j++){
         cout << graph[i][j] << " ";
```

```
        }
        cout << endl;
    }
}
```

// 22. Garph Traversal (Deapth First Search-DFS).cpp

```
#include<bits/stdc++.h>
using namespace std;
template<typename tp>
class graph{
   map<tp,list<tp>> mp;
public:
   void addEdge(int x, int y){
      mp[x].push_back(y);
      mp[y].push_back(x);
   }
   void DFS_help(tp src, map<tp,bool> &visited){
      cout << src << " ";
      visited[src] = true;
      for(tp d:mp[src]){
         if(!visited[d]){
            DFS_help(d, visited);
         }
      }
   }
   void DFS(tp src){
      map<tp,bool> visited;
```

```
        for(auto d:mp){
            tp node = d.first;
            visited[node] = false;
        }
        DFS_help(src,visited);
    }
};
int main(){
    graph<int> g;
    g.addEdge(0, 1);
    g.addEdge(1, 2);
    g.addEdge(2, 3);
    g.addEdge(3, 4);
    g.addEdge(4, 5);
    g.addEdge(3, 0);

    g.DFS(0);
}
```

//18. Heap Sort(Max heap).cpp

```
#include<bits/stdc++.h>
using namespace std;
void heapify(int arr[],int n,int root){
   int maxi = root;
   int left = 2*root;
   int right = 2*root + 1;
   if(left < n && arr[left] > arr[maxi]) maxi = left;
   if(right < n && arr[right] > arr[maxi]) maxi = right;

   if(maxi != root){
      swap(arr[root],arr[maxi]);
      heapify(arr,n,maxi);
   }
}
void heapSort(int arr[],int n){
   for(int i = 0; i < n; i++)  heapify(arr,n,i);

   for(int i = n-1; i >= 0; i--){
      swap(arr[0],arr[i]);
      heapify(arr,i,0);
   }
```

```
}
int main(){
   cout << "Enter number of heap elements: ";
   int n;  cin >> n;
   int arr[n];//4,17,3,12,9,6
   for(int i=0; i<n; i++) cin >> arr[i];
   cout << "\nBefore heap sort!!\n";
   for(int d:arr) cout << d << " ";

   heapSort(arr,n);

   cout << "\nAfter heap sort!!\n";
   for(int d:arr) cout << d << " ";

}
```

//18. Heap Sort(Min heap).cpp

```
#include<bits/stdc++.h>
using namespace std;
void heapify(int arr[],int n,int root){
   int mini = root;
   int left = 2*root+1;
   int right = 2*root + 2;
   if(left < n && arr[left] < arr[mini]) mini = left;
   if(right < n && arr[right] < arr[mini]) mini = right;

   if(mini != root){
      swap(arr[root],arr[mini]);
      heapify(arr,n,mini);
   }
}
void heapSort(int arr[],int n){
   for(int i = 0; i < n; i++)  heapify(arr,n,i);

   for(int i=n-1; i >= 0; i--){
      swap(arr[0],arr[i]);
      heapify(arr,i,0);
   }
```

```
}
int main(){
   cout << "Enter number of heap elements: ";
   int n;  cin >> n;
   int arr[n];//4,17,3,12,9,6
   for(int i=0; i<n; i++) cin >> arr[i];
   cout << "\nBefore heap sort!!\n";
   for(int d:arr) cout << d << " ";

   heapSort(arr,n);

   cout << "\nAfter heap sort!!\n";
   for(int i = n-1; i>=0; i--) cout << arr[i] << " ";

}
```