Question number 1:-

#include<iostream>

#include<stdlib.h>

#include<time.h>

using namespace std;

class Game {

int Cost;

char Arr[15][15];

int s, s1;

public:

Game() {

Cost = s = s1 = 0;

for (int i = 0; i < 15; i++) {

for (int j = 0; j < 15; j++) {

Arr[i][j] = ' ';

}

}

srand(time(0));

int x,y;

for (int i = 0; i < 50; i++) {

while (true) {

x = rand() % 14;

y = rand() % 14;

if (x != 0 && y != 0)

break;

}

Arr[x][y] = 'X';

}

Arr[0][0] = 'S';

Arr[14][14] = 'E';

}

int getcost() {

return Cost;

}

void disp() {

for (int i = 14; i >= 0; i--) {

for (int j = 0; j < 15; j++) {

cout << Arr[i][j] << " | ";

}

cout << endl;

}

}

bool Upward() {

Arr[s][s1] = ' ';

s = s + 1;

if (s > 14) {

s = s - 14;

}

if (Arr[s][s1] == 'X') {

return 0;

}

Arr[s][s1] = 'S';

Cost = Cost + 2;

return 1;

}

bool Right() {

Arr[s][s1] = ' ';

s1 = s1 + 1;

if (s1 > 14) {

s1 = s1 - 14;

}

if (Arr[s][s1] == 'X') {

return 0;

}

Arr[s][s1] = 'S';

Cost = Cost + 2;

return 1;

}

bool Dignal() {

Arr[s][s1] = ' ';

s1 = s1 + 1;

s = s + 1;

if (s > 14) {

s = s - 14;

}

if (s1 > 14) {

s1 = s1 - 14;

}

if (Arr[s][s1] == 'X') {

return 0;

}

Arr[s][s1] = 'S';

Cost = Cost + 3;

return 1;

}

bool Comp() {

if (s == 14 && s1 == 14) {

return 1;

}

return 0;

}

};

int main() {

int x;

bool fg;

Game obj;

while (!obj.Comp()) {

system("cls");

obj.disp();

cout << endl << "Press 1 to move Upward....." << endl;

cout << "Press 2 to move Right....." << endl;

cout << "Press 3 to move dignally....." << endl;

cout << "Enter : ";

cin >> x;

switch (x)

{

case 1: {

fg = obj.Upward();

if (fg == 0) {

system("cls");

cout << "You hit the Wall....!" << endl;

cout << "Game is over" << endl;

system("pause");

return 0;

}

break;

}

case 2:

fg=obj.Right();

if (fg == 0) {

system("cls");

cout << "You hit the Wall....!" << endl;

cout << "Game is over" << endl;

system("pause");

return 0;

}

break;

case 3:

fg=obj.Dignal();

if (fg == 0) {

system("cls");

cout << "You hit the Wall....!" << endl;

cout << "Game is over" << endl;

system("pause");

return 0;

}

break;

default:

cout << "Wrong input......" << endl;

break;

}

}

system("cls");

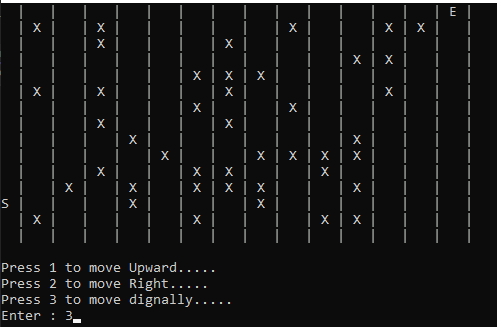
cout << "You have won...." << endl;

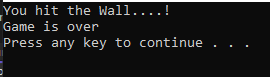
cout << "Total cost : " << obj.getcost() << endl;

system("pause");

}

Result :-





Question number 2(B)1 :-

#include<iostream>

using namespace std;

class Ad\_Matrix {

int nodes = 8;

int Arr[8][8];

public:

Ad\_Matrix() {

for (int i = 0; i < nodes; i++)

{

for (int j = 0; j < nodes; j++)

{

Arr[i][j] = 0;

}

}

}

void setval()

{

bool flag;

int x, y;

for (int i = 0; i < 8; i++)

{

flag = true;

while (flag)

{

system("cls");

cout << endl << "Total Nodes : " << nodes << endl;

cout << "\nEdge (" << i + 1 << ")\n";

cout << "1st Node : ";

cin >> x;

cout << "2st Node : ";

cin >> y;

if (x >= 0 && x <= 8 && y >= 0 && y <= 8)

{

Arr[x - 1][y - 1] = 1;

Arr[y - 1][x - 1] = 1;

flag = false;

}

else

{

cout << "\nWrong Input! enter Again." << endl;

}

}

}

}

void display()

{

cout << "\nEntered Edges Is A={";

for (int i = 0; i < 8; i++)

{

for (int j = 0; j < 8; j++)

{

if (Arr[i][j] == 1)

cout << "(" << i + 1 << "," << j + 1 << ")";

if (j == 8 - 1 && i == 8 - 1)

{

cout << "}";

break;

}

if (Arr[i][j] == 1)

cout << ",";

}

}

cout << "\n\nAdjacency Matrix:\n";

cout << " 1 2 3 4 5 6 7 8" << endl;

for (int i = 0; i < 8; i++)

{

cout << i + 1 << " ";

for (int j = 0; j < 8; j++)

{

cout << Arr[i][j] << " ";

}

cout << endl;

}

}

};

int main()

{

Ad\_Matrix obj;

cout << "Enter Edges:\n";

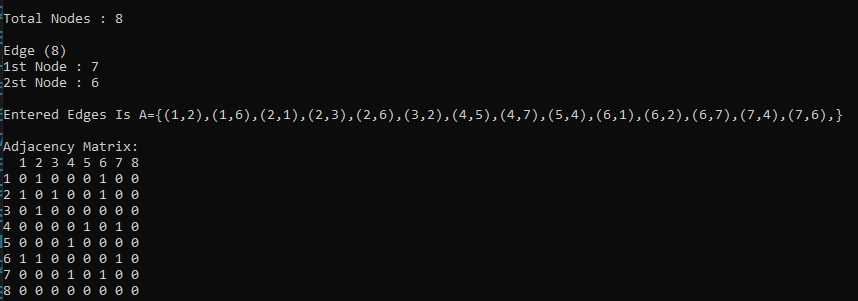
obj.setval();

obj.display();

return 0;

}

Result :-



Question number 2(B)2 :-

#include<iostream>

using namespace std;

class Ad\_Matrix {

int nodes = 8;

int Arr[8][8];

public:

Ad\_Matrix() {

for (int i = 0; i < nodes; i++)

{

for (int j = 0; j < nodes; j++)

{

Arr[i][j] = 0;

}

}

}

void setval()

{

bool flag;

int x, y;

for (int i = 0; i < 8; i++)

{

flag = true;

while (flag)

{

system("cls");

cout << endl << "Total Nodes : " << nodes << endl;

cout << "\nEdge (" << i + 1 << ")\n";

cout << "1st Node : ";

cin >> x;

cout << "2st Node : ";

cin >> y;

if (x >= 0 && x <= 8 && y >= 0 && y <= 8)

{

Arr[x - 1][y - 1] = 1;

Arr[y - 1][x - 1] = 1;

flag = false;

}

else

{

cout << "\nWrong Input! enter Again." << endl;

}

}

}

}

void disp\_Matrix()

{

cout << "\nEntered Edges Is A={";

for (int i = 0; i < 8; i++)

{

for (int j = 0; j < 8; j++)

{

if (Arr[i][j] == 1)

cout << "(" << i + 1 << "," << j + 1 << ")";

if (j == 8 - 1 && i == 8 - 1)

{

cout << "}";

break;

}

if (Arr[i][j] == 1)

cout << ",";

}

}

cout << "\n\nAdjacency Matrix:\n";

cout << " 1 2 3 4 5 6 7 8" << endl;

for (int i = 0; i < 8; i++)

{

cout << i + 1 << " ";

for (int j = 0; j < 8; j++)

{

cout << Arr[i][j] << " ";

}

cout << endl;

}

}

void disp\_List()

{

cout << "\nAdjancy List Given Below:\n";

for (int i = 0; i < 8; i++)

{

cout << i + 1;

for (int j = 0; j < 8; j++)

{

if (Arr[i][j] == 1)

{

cout << " -> " << j + 1;

}

}

cout << endl;

}

}

};

int main()

{

Ad\_Matrix obj;

cout << "Enter Edges:\n";

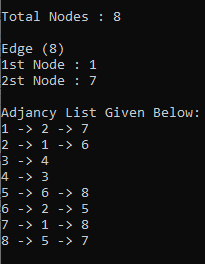
obj.setval();

obj.disp\_List();

return 0;

}

Result :-



Question number 2(A) :-

#include<iostream>

using namespace std;

class Ad\_Matrix {

int nodes = 6;

int Arr[6][6];

char A[6];

int A1[6];

public:

int Ret(char x) {

for (int i = 0; i < 6; i++) {

if (A[i] == x)

return i;

}

}

Ad\_Matrix() {

for (int i = 0; i < nodes; i++)

{

for (int j = 0; j < nodes; j++)

{

Arr[i][j] = 0;

}

}

char x = 'a';

for (int i = 0; i < 6; i++) {

A1[i] = i;

A[i] = x;

x++;

}

}

void setval()

{

int weight;

bool flag;

char x, y;

for (int i = 0; i < 9; i++)

{

flag = true;

while (flag)

{

system("cls");

cout << endl << "Total Nodes : " << nodes << endl;

cout << "\nEdge (" << i + 1 << ")\n";

cout << "1st Node : ";

cin >> x;

cout << "2st Node : ";

cin >> y;

cout << "Enter weight : ";

cin >> weight;

Arr[Ret(x)][Ret(y)] = weight;

Arr[Ret(y)][Ret(x)] = weight;

flag = false;

}

}

}

void disp\_Matrix()

{

cout << "\n\nAdjacency Matrix:\n";

cout << ' ';

for (int i = 0; i < 6; i++) {

cout << " " << A[i];

}

cout << endl;

for (int i = 0; i < nodes; i++)

{

cout << A[i] << " ";

for (int j = 0; j < nodes; j++)

{

cout << Arr[i][j] << " ";

}

cout << endl;

}

}

void disp\_List()

{

cout << "\nAdjancy List Given Below:\n";

for (int i = 0; i < nodes; i++)

{

cout << i + 1;

for (int j = 0; j < nodes; j++)

{

if (Arr[i][j] == 1)

{

cout << " -> " << j + 1;

}

}

cout << endl;

}

}

};

int main()

{

Ad\_Matrix obj;

cout << "Enter Edges:\n";

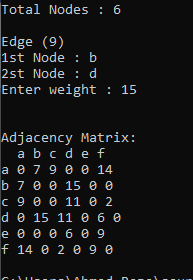
obj.setval();

obj.disp\_Matrix();

return 0;

}

Result :-



Question Number 4 :-

#include<iostream>

using namespace std;

class Heap

{

int\* arr;

int cap;

int size;

public:

int getSize() {

return size;

}

int getCap() {

return cap;

}

Heap(int c)

{

size = 0;

cap = c;

arr = new int[cap];

cout << "Heap Is Created." << endl;

}

void print()

{

cout << endl;

for (int i = 0; i < size; i++)

cout << arr[i] << " ";

cout << endl;

}

int parent(int i)

{

return ((i - 1) / 2);

}

int left\_child(int i)

{

return ((2 \* i) + 1);

}

int Right\_child(int i)

{

return ((2 \* i) + 2);

}

void swap(int& arr, int& a)

{

int num;

num = arr;

arr = a;

a = num;

}

void insert(int val)

{

if (size == cap)

{

cout << "OverFlow." << endl;

return;

}

size++;

int temp = size - 1;

arr[temp] = val;

while (temp != 0 && arr[parent(temp)] > arr[temp])

{

swap(arr[temp], arr[parent(temp)]);

temp = parent(temp);

}

}

void SwapL(int num, int newval)

{

arr[num] = newval;

while (num != 0 && arr[parent(num)] > arr[num])

{

swap(arr[num], arr[parent(num)]);

num = parent(num);

}

}

int Rearrange()

{

int num;

if (size <= 0)

return INT\_MIN;

if (size == 1)

{

size--;

return arr[0];

}

num = arr[0];

arr[0] = arr[size - 1];

size--;

MinHeapify(0);

return num;

}

void MinHeapify(int val)

{

int l\_val = left\_child(val);

int r\_val = Right\_child(val);

int smallest = val;

if (l\_val < size && arr[l\_val] < arr[val])

smallest = l\_val;

if (r\_val < size && arr[r\_val] < arr[smallest])

smallest = r\_val;

if (smallest != val)

{

swap(arr[val], arr[smallest]);

MinHeapify(smallest);

}

}

void deletekey(int num)

{

int min = -1;

SwapL(num, min);

Rearrange();

cout << "Smallest Number Is Deleted In Heap : " << endl;

}

};

int main()

{

int size, data, con;

cout << "Enter Size Of Heap : ";

cin >> size;

Heap obj(size);

do

{

cout << "Press 1 to Insert " << endl;

cout << "Press 2 to delete " << endl;

cout << "Press 3 to display " << endl;

cout << "Press 0 to Exit " << endl;

cout << "Size Of Heap : " << obj.getSize() << endl;

cout << "Capacity of Heap : " << obj.getCap() << endl;

cin >> con;

switch (con)

{

case 1:

{

cout << "\nEnter Value To Insertion In Heap : ";

cin >> data;

obj.insert(data);

cout << endl;

system("cls");

break;

}

case 2:

{

obj.deletekey(0);

cout << endl;

system("pause");

system("cls");

break;

}

case 3:

{

cout << endl;

obj.print();

cout << endl;

system("pause");

system("cls");

break;

}

case 0:

break;

default:

cout << "\nWrong Input!------- Try Again--------" << endl;

cout << endl;

system("cls");

}

} while (true);

return 0;

}

Result :-

