ARTIFICIAL INTELLIGENCE

Submitted by: **Submitted to:**

Malaika   Mr. Abhishek Bhardwaj
1803545   [A.P CSE/IT]
[B.tech/CSE-7TH]
CT-IEMT

TABLE OF CONTENTS

PRACTICAL:1

AIM: WAP to implement Tower of Hanoi problem in C++.

INPUT:

```
#include <iostream>
using namespace std;
void toh(int ,char,char,char);
int main()
{
char a,b,c;
int x;
a='A';
b='B';
c='C';
cout<<"Enter value of discs";
cin>>x;
toh(x,a,c,b);
return 0;
}
void toh(int x,char a ,char c,char b)
{
if(x==1)
{
        cout<<"Move the disc 1 from "<<a<<" to "<<c<<endl;
        x=x-1;
}
else
{
        toh(x-1,a,b,c) ;
        cout<<"Move the disc "<<x<<" from"<<a<<" to "<<c<<endl;
        toh(x-1,b,c,a);
}}
```

OUTPUT:

```
Enter value of discs3
Move the disc 1 from A to C
Move the disc 2 fromA to B
Move the disc 1 from C to B
Move the disc 3 fromA to C
Move the disc 1 from B to A
Move the disc 2 fromB to C
Move the disc 1 from A to C

Process returned 0 (0x0)   execution time : 2.557 s
Press any key to continue.
```

PRACTICAL:2

AIM: WAP to implement tic-tac-toe problem in C++.

INPUT:

```cpp
#include <iostream>
using namespace std;
char square[9] = {'0','1','2','3','4','5','6','7','8'};
int checkwin()
{
        if (square[0] == square [1]  && square[1] == square[2] )
        {       if ( square [0] == 'X' )
                return 1;
                else
                return 2;
        }
        else
        if (square[3] == square [4]  && square[4] == square[5] )
                {       if ( square [3] == 'X' )
                return 1;
                else
                return 2;
        }
        else
        if (square[6] == square [7]  && square[7] == square[8] )
                {       if ( square [6] == 'X' )
                return 1;
                else
                return 2;
        }
        else
        if (square[0] == square [3]  && square[3] == square[6] )
                {       if ( square [0] == 'X' )
                return 1;
                else
                return 2;
        }
        else
        if (square[1] == square [4]  && square[4] == square[7] )
                {       if ( square [1] == 'X' )
                return 1;
                else
                return 2;
        }
    else
        if (square[2] == square [5]  && square[5] == square[8] )
                {       if ( square [2] == 'X' )
```

```cpp
                        return 1;
                        else
                        return 2;
                }
else
                if (square[0] == square [4]  && square[4] == square[8] )
                        {        if ( square [0] == 'X' )
                        return 1;
                        else
                        return 2;
                }
        else
                if (square[2] == square [4]  && square[4] == square[6] )
                        {        if ( square [2] == 'X' )
                        return 1;
                        else
                        return 2;
                }
        else
                if (square[0] == square [3]  && square[3] == square[6] )
                        {        if ( square [0] == 'X' )
                        return 1;
                        else
                        return 2;
                }
        else
                return 0;
}
void mark(int player, int box)
{
        if (player == 1 )
        {
                square[box] = 'X';
        }
        else
                square[box] = 'Y';
}
void display()
{
                for(int i=0;i<9;i++)
                {
                        cout<< square[i] << "\t" ;
                                if (i == 2 || i== 5 || i==8)
                                        cout<<"\n";
}
```

```cpp
}
int main()
{
            int player1 = 1, player2 =2 ;
            int box, result = 0, flag = 0;
            for(int i=1;i<5;i++)
    {
            cout<< "\n Player " << player1 << "Enter the Box";
            cin>> box;
            mark( player1, box);
            display();
            result =checkwin();
            if (result == 1 )
            {       cout<<"\n Congratualtions! player " << player1 << " has Won ";
                    flag = 1;
                    break;

            }
            else
            if (result == 2 )
            {       cout<<"\n Congratualtions! player " << player2 << " has Won ";
                    flag = 1;
                    break;

            }
            cout<< "\n Player " << player2 << "Enter the Box";
            cin>> box;
            mark ( player2, box);
            display();
            result =checkwin();
            if (result == 1 )
            {       cout<<"\n Congratualtions! player " << player1 << " has Won ";
                    flag = 1;
                    break;

            }
            else
            if (result == 2 )
            {       cout<<"\n Congratualtions! player " << player2 << " has Won ";
                    flag = 1;
                    break;

            }
    }
            if (flag == 0 )
            cout<<" \n Sorry, The game is a draw ";
      return 0;
}
```

OUTPUT:

```
 Player 1Enter the Box0
X       1       2
3       4       5
6       7       8

 Player 2Enter the Box4
X       1       2
3       Y       5
6       7       8

 Player 1Enter the Box3
X       1       2
X       Y       5
6       7       8

 Player 2Enter the Box1
X       Y       2
X       Y       5
6       7       8

 Player 1Enter the Box6
X       Y       2
X       Y       5
X       7       8

 Congratualtions! player 1 has Won
Process returned 0 (0x0)   execution time : 69.314 s
Press any key to continue.
```

PRACTICAL: 3

AIM: WAP to implement water jug problem in C++.

INPUT:

```cpp
#include<iostream.h>
#include<iomanip.h>
#include<math.h>
#include<conio.h>
int xcapacity;
int ycapacity;
void display(int a, int b,int s);
int min(int d, int f)
{if (d < f)
return d;
else
return f;
} int steps(int n)
{
int x = 0, y = 0, step = 0;
int temp;
cout << setw(55) << " Vessel A Vessel B Steps" << endl; while
(x != n )
{if (x == 0)
{ x = xcapacity;
step += 1;
cout << "Fill X   "; display(x, y,step);
} else if (y ==
ycapacity) { y = 0;
step++;
cout << "Empty Y    "; display(x, y,step);
}else {
temp = min(ycapacity – y, x);
y = y + temp;
x = x – temp;
step++;
cout << "Pour X in Y"; display(x, y, step);
} }
return step;
}void display(int a, int b,int s)
{cout << setw(16) << a << setw(15) << b << setw(15)<<s<<endl;
```

```
} void main()
{
int n, ans;
clrscr();
cout << "Enter the liters(GOAL) of water required to be filled in Vessel
1:"; cin >> n;
cout << "Enter the capacity of the vessel:
"; cin >> xcapacity;
cout << "Enter the capacity of the second 7essel: ";
cin >> ycapacity;
ans = steps(n);
cout << "Steps Required: " <<
ans; cout<<"pause";
}
```
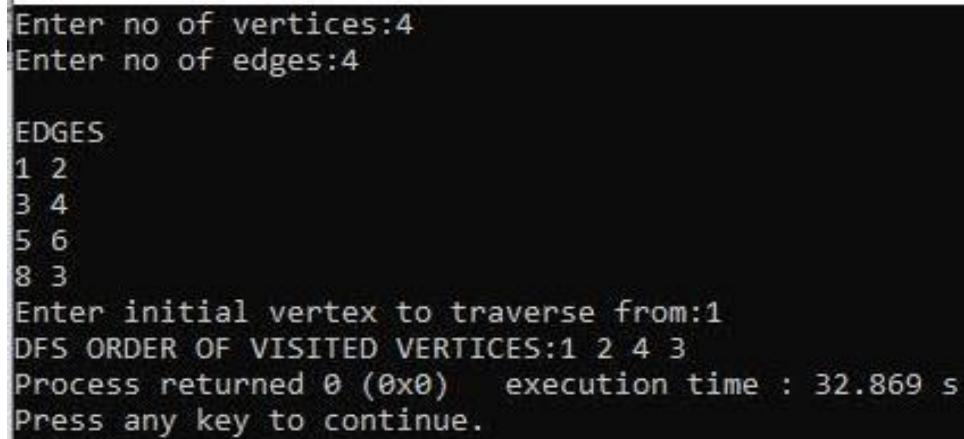
OUTPUT:

```
Enter the liters(GOAL) of water required to be filled in Vessel 1:2
Enter the capacity of the vessel: 4
Enter the capacity of the second vesssel: 3
                        Vessel A    Vessel B    Steps
Fill X                  4           0           1
Pour X in Y             1           3           2
Empty Y                 1           0           3
Pour X in Y             0           1           4
Fill X                  4           1           5
Pour X in Y             2           3           6
Steps Required: 6 pause_
```

PRACTICAL: 4

AIM: WAP to implement DFS in C++.

INPUT:

```cpp
#include<iostream>
#include<conio.h>
#include<stdlib.h>
int cost[10][10],i,j,k,n,stk[10],top,v,visit[10],visited[10];
int main()
{
    int m;
    cout <<"Enter no of vertices:";
    cin >> n;
    cout <<"Enter no of edges:";
    cin >> m;
    cout <<"\nEDGES \n";
    for(k=1; k<=m; k++)
    {
        cin >>i>>j;
        cost[i][j]=1;
    }
    cout <<"Enter initial vertex to traverse from:";
    cin >>v;
    cout <<"DFS ORDER OF VISITED VERTICES:";
    cout << v <<" ";
    visited[v]=1;
    k=1;
    while(k<n)
    {
        for(j=n; j>=1; j--)
            if(cost[v][j]!=0 && visited[j]!=1 && visit[j]!=1)
            {
                visit[j]=1;
                stk[top]=j;
                top++;
            }
        v=stk[--top];
        cout<<v << " ";
        k++;
        visit[v]=0;
```

```
        visited[v]=1;
    }
    return 0;
}
```

OUTPUT:

```
Enter no of vertices:4
Enter no of edges:4

EDGES
1 2
3 4
5 6
8 3
Enter initial vertex to traverse from:1
DFS ORDER OF VISITED VERTICES:1 2 4 3
Process returned 0 (0x0)    execution time : 32.869 s
Press any key to continue.
```
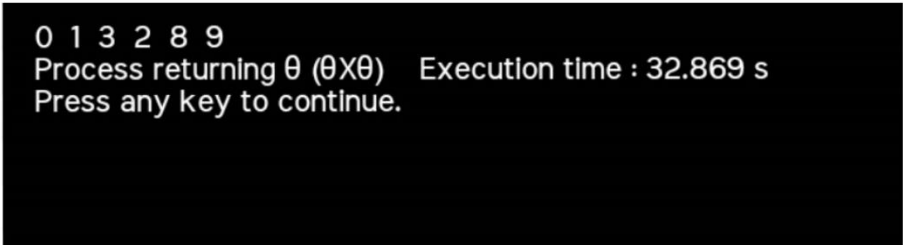
PRACTICAL: 5

AIM: WAP to implement Best First Search in C++.

INPUT:

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef pair<int, int> pi;
vector<vector<pi> > graph;
void addedge(int x, int y, int cost)
{
    graph[x].push_back(make_pair(cost, y));
    graph[y].push_back(make_pair(cost, x));
}
void best_first_search(int source, int target, int n)
{
    vector<bool> visited(n, false);
    priority_queue<pi, vector<pi>, greater<pi> > pq;
    pq.push(make_pair(0, source));
    visited = true;
    while (!pq.empty()) {
        int x = pq.top().second;
        cout << x << " ";
        pq.pop();
        if (x == target)
            break;
        for (int i = 0; i < graph[x].size(); i++) {
            if (!visited[graph[x][i].second]) {
                visited[graph[x][i].second] = true;
                pq.push(graph[x][i]);
            }
        }
    }
}
int main()
{
    int v = 14;
    graph.resize(v);
    addedge(0, 1, 3);
    addedge(0, 2, 6);
    addedge(0, 3, 5);
```

```
    addedge(1, 4, 9);
    addedge(1, 5, 8);
    addedge(2, 6, 12);
    addedge(2, 7, 14);
    addedge(3, 8, 7);
    addedge(8, 9, 5);
    addedge(8, 10, 6);
    addedge(9, 11, 1);
    addedge(9, 12, 10);
    addedge(9, 13, 2);
    int source = 0;
    int target = 9;
    best_first_search(source, target, v);
    return 0;
}
```

OUTPUT:


```
0 1 3 2 8 9
Process returning θ (θXθ)    Execution time : 32.869 s
Press any key to continue.
```

PRACTICAL: 6

AIM: WAP to implement 3*3 puzzle in C++.

INPUT:

```cpp
#include <bits/stdc++.h>
using namespace std;
#define N 3
struct Node
{
    Node* parent;
    int mat[N][N];
    int x, y;
    int cost;
    int level;
};
int printMatrix(int mat[N][N])
{
    for (int i = 0; i < N; i++)
    {
        for (int j = 0; j < N; j++)
            printf("%d ", mat[i][j]);
        printf("\n");
    }
}
Node* newNode(int mat[N][N], int x, int y, int newX,
        int newY, int level, Node* parent)
{
    Node* node = new Node;
    node->parent = parent;
    memcpy(node->mat, mat, sizeof node->mat);
    swap(node->mat[x][y], node->mat[newX][newY]);
    node->cost = INT_MAX;
    node->level = level;
    node->x = newX;
    node->y = newY;
    return node;
}
int row[] = { 1, 0, -1, 0 };
int col[] = { 0, -1, 0, 1 };
int calculateCost(int initial[N][N], int final[N][N])
```

```
{
   int count = 0;
   for (int i = 0; i < N; i++)
     for (int j = 0; j < N; j++)
       if (initial[i][j] && initial[i][j] != final[i][j])
          count++;
   return count;
}
int isSafe(int x, int y)
{
   return (x >= 0 && x < N && y >= 0 && y < N);
}
void printPath(Node* root)
{
   if (root == NULL)
      return;
   printPath(root->parent);
   printMatrix(root->mat);
   printf("\n");
}
struct comp
{
   bool operator()(const Node* lhs, const Node* rhs) const
   {
      return (lhs->cost + lhs->level) > (rhs->cost + rhs->level);
   }
};
void solve(int initial[N][N], int x, int y,
        int final[N][N])
{
   priority_queue<Node*, std::vector<Node*>, comp> pq;
   Node* root = newNode(initial, x, y, x, y, 0, NULL);
   root->cost = calculateCost(initial, final);
   pq.push(root);
   while (!pq.empty())
   {
      Node* min = pq.top();
      pq.pop();
      if (min->cost == 0)
```

```
    {
      printPath(min);
      return;
    }
    for (int i = 0; i < 4; i++)
    {
      if (isSafe(min->x + row[i], min->y + col[i]))
      {
        Node* child = newNode(min->mat, min->x, min->y, min->x + row[i],

  min->y + col[i],                    min->level + 1, min);
          child->cost = calculateCost(child->mat, final);
        pq.push(child);
      }
    }
  }
}
int main()
{
  int initial[N][N] =
  {
    {1, 2, 3},
    {5, 6, 0},
    {7, 8, 4}
  };
  int final[N][N] =
  {
    {1, 2, 3},
    {5, 8, 6},
    {0, 7, 4}
  };
  int x = 1, y = 2;
  solve(initial, x, y, final);
  return 0;
}
```

OUTPUT:

```
1 2 3
5 6 0
7 8 4

1 2 3
5 0 6
7 8 4

1 2 3
5 8 6
7 0 4

1 2 3
5 8 6
0 7 4
Process returning θ (θXθ) Execution time :35.786 s
Press any key to continue
```

PRACTICAL: 7

AIM: WAP to implement a heuristic search procedure in C++.
INPUT:

```cpp
#include<bits/stdc++.h>
using namespace std;
vector<vector<int> > graph;
bool vis[100011];
int i,j;
vector<int> solve_vertex(int n,int e)
{
        vector<int> S;
        for(i=0;i<n;i++)
        {
                if(!vis[i])
                {
                        for(j=0;j<(int)graph[i].size();j++)
                        {
                                if(!vis[graph[i][j]])
                                {
                                        vis[i]=true;
                                        vis[graph[i][j]]=true;
                                        break;
                                }
                        }
                }
        }
        for(i=0;i<n;i++)
                if(vis[i])
                        S.push_back(i);
        return S;
}
int main()
{
        int n,e,x,y;
        cout<<"Enter number of vertices:";
        cin>>n;
        cout<<"Enter number of Edges:";
        cin>>e;
        graph.resize(n);
```

```
        memset(vis,0,sizeof(vis));
        for(i=0;i<e;i++)
        {
                cout<<"Enter the end-points of edge "<<i+1<<" : ";
                cin>>x>>y;
                x--; y--;
                graph[x].push_back(y);
                graph[y].push_back(x);
        }
        vector<int> S = solve_vertex(n,e);
        cout<<"The required vertex cover is as follows:\n";
        for(i=0;i<(int)S.size();i++)
        cout<<S[i]+1<<" ";
        return 0;
}
```

OUTPUT:

```
Enter number of vertices:4
Enter number of Edges:5
Enter the end-points of edge 1 : 2 1
Enter the end-points of edge 2 : 3 2
Enter the end-points of edge 3 : 4 3
Enter the end-points of edge 4 : 1 4
Enter the end-points of edge 5 : 1 3
The required vertex cover is as follows:
1 2 3 4
Process returning θ (θXθ) Execution time :35.786 s
Press any key to continue
```

PRACTICAL: 8

AIM: WAP to implement a Production System in C++.

INPUT:

```
#include<iostream.h>
#include<conio.h>
int main()
{
char answer;
clrscr();
cout<<"Answer the following question to determine whether JOHN should get scholarship or not?"<<endl;
cout<<"Q1) Is John a Student?(y/n)\n";
cin>>answer;
if(answer=='y'){
cout<<"John enjoys Student Life"<<endl;
cout<<"John Enjoys Student Life --> John Meets Friends"<<endl; cout<<"John Enjoys Meets Friends --> John Needs Money"<<endl; cout<<"Q2) Does John has a job?(y/n)";
cin>>answer;
if(answer=='y')
{
cout<<"John Has Job --> John Has Free Time"<<endl;
cout<<"Since John Works in Free Time--> John Is Not Not Good In Studies "<<endl;
cout<<"John should not get the scholarship";
}
}
else
cout<<"John Will not recieve scholarship as he is not a student"<<endl;
return 0;
}
```

OUTPUT:

PRACTICAL: 9

AIM: WAP to implement An Expert System in C++.

INPUT:

```cpp
#include <iostream>
using namespace std;
void measels(char,char,char,char,char);
void flu(char,char,char,char,char,char,char,char);
void
cold(char,char,char,char,char);
void
chickenpox(char,char,char,char);
int main()
{
  char name[50];
  char a,b,c,d,e,f,g,h,i,j,k;
  cout << "Please enter your name.. " << endl;
  cin>> name;
  cout << "Do you have fever? (y/n)"<< endl;
  cin>>a;
  cout << "Do you have rashes? (y/n)"<< endl;
  cin>>b;
  cout << "Do you have headache? (y/n)"<< endl;
  cin>>c;
  cout << "Do you have running nose? (y/n)"<<
  endl; cin>>d;
  cout << "Do you have conjunctivities? (y/n)"<<
  endl; cin>>e;
  cout << "Do you have cough? (y/n)"<< endl;
  cin>>f;
  cout << "Do you have ache? (y/n)"<< endl;
  cin>>g;
  cout << "Do you have chills? (y/n)"<< endl;
  cin>>h;
  cout << "Do you have swollen glands? (y/n)"<<
  endl; cin>>i;
  cout << "Do you have snezzing? (y/n)"<< endl;
  cin>>j;
  cout << "Do you have sore throat? (y/n)"<< endl;
  cin>>k;
  measels(a,f,e,d,
  b);
  flu(a,c,g,e,h,k,f,
```

```
    d);
    cold(c,j,k,d,h);
    chickenpox(a,h,
    g,b); return 0;
     } void measels(char q,char w,char r,char
                  t,char y)
 {
   if(q=='y'&&w=='y'&& r=='y' && t=='y' &&
                      y==
      'y') cout<< "You may have measels."<< endl;
      else
      cout<
      < "";
 }
 void flu(char q,char w,char r,char t,char y,char p,char l,char x)
 {
    if(q=='y'&&w=='y'&& r=='y' && t=='y' && y== 'y'&& p=='y' && l=='y' &&
      x=='y') cout<< "You may have flu."<< endl;
      else
      cout<
      < "";}
```

OUTPUT:

```
Please enter your name..
Rajwinder
Do you have fever? (y/n)
y
Do you have rashes? (y/n)
y
Do you have headache? (y/n)
n
Do you have running nose? (y/n)
n
Do you have conjunctivities? (y/n)
y
Do you have cough? (y/n)
n
Do you have ache? (y/n)
y
Do you have chills? (y/n)
y
Do you have swallow glands? (y/n)
n
Do you have sneezing? (y/n)
n
Do you have sore throat? (y/n)
n
You may have chicken-pox.
Process returning θ (θXθ) Execution time :35.786 s
Press any key to continue
```

PRACTICAL: 10

AIM: WAP to implement A*Algorithm in C++.

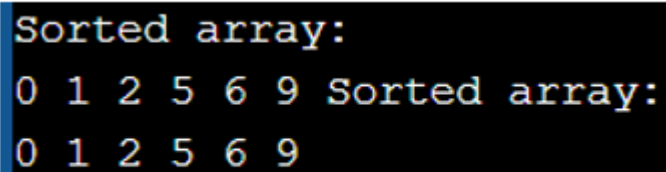INPUT:

```cpp
#include <iostream>

void swap (int *p1, int *p2)

{

int temp = *p1;

*p1 = *p2;

*p2 = temp;

}

void bSort(int arrnumbers[], int n)

{

int i, j; bool check;

for (i = 0; i < n-1; i++)

{

check = false;

for (j = 0; j < n-i-1; j++)

{

if (arrnumbers[j] > arrnumbers[j+1])

{

swap(&arrnumbers[j], &arrnumbers[j+1]); check = true;

}}

if (check == false) break;

}}

void print(int arrnumbers[], int sizeofarray)

{

int i;

for (i=0; i < sizeofarray; i++) printf("%d ", arrnumbers[i]);
```

}

int main()

{

int arrnumbers[] = {5, 6, 1, 0, 2, 9};

int n = sizeof(arrnumbers)/sizeof(arrnumbers[0]); bSort(arrnumbers, n);

printf("Sorted array: \n"); print(arrnumbers, n); return 0;

}

OUTPUT:

```
Sorted array:
0 1 2 5 6 9 Sorted array:
0 1 2 5 6 9
```