Experiment No : 3

Aim : Write a C/C++/Java program to solve the above 8-puzzle problem using Steepest Ascent Hill Climbing Algorithm. Select an appropriate evaluation function to rate the states.

Theory : The 8-puzzle is a square tray in which are placed, eight square tiles. The remaining ninth square is uncovered. Each tile has a number on it. A tile that is adjacent to the blank space can be slid into that space. A game consists of a starting position and a specified goal position. The goal is to transform the starting position into the goal position by sliding the tiles around.

Procedure : Formalize the above problem in terms of state-space search. Consider the following Initial and Goal States.

```
Start State:      Goal State:
_ _ _             _ _ _
|2|8|3|            |1|2|3|
|1|6|4|            |8|  |4|
|7|  |5|           |7|6|5|
_ _ _             _ _ _
```

Evaluation Function Used :
The function used to evaluate the best state to move to was checking how many squares were in the correct position as per the goal state.

Code :

```
#include<iostream>
#include<bits/stdc++.h>
using namespace std;
string inSquare={‘2’,’8’,’3’,’1’,’6’,’4’,’7’,’ ’,’5’};
string goalState={‘1’,’2’,’3’,’8’,’ ’,’4’,’7’,’6’,’5’};
bool reachedGoal=false;
map<string,bool> visited;
void printSquare(string sq){
int i,j,idx=0;
cout<<" _ _ _"<<endl;
for(i=0;i<3;i++){
cout<<"|";
for(j=0;j<3;j++){
cout<<sq[idx]<<"|";
idx++;
}
cout<<endl;
}
cout<<" - - -"<<endl;
}
int scoreAtNode(string sq){
//Score is number of tiles in the right position
int value=0,i;
for(i=0;i<9;i++){
if(sq[i]==goalState[i])
value++;
}
return value;
}
list<string> getSuccessors(string sq){
list<string> neighbours;
// Get index of the space
int spaceIdx=sq.find(" ");
// Find next possible moves
// move down the tile above the space
if(spaceIdx-3>=0){
string next=sq;
swap(next[spaceIdx],next[spaceIdx-3]);
if(visited.count(next)==0)
neighbours.push_back(next);
}
//move up the tile below the space
if(spaceIdx+3<9){
string next=sq;
swap(next[spaceIdx],next[spaceIdx+3]);
if(visited.count(next)==0)
neighbours.push_back(next);
}
//move the tile to the left of the space
if(spaceIdx%3!=0){
string next=sq;
swap(next[spaceIdx],next[spaceIdx-1]);
if(visited.count(next)==0)
neighbours.push_back(next);
}
//move the tile to the right of the square
if((spaceIdx+1)%3!=0){
string next=sq;
swap(next[spaceIdx],next[spaceIdx+1]);
if(visited.count(next)==0)
neighbours.push_back(next);
}
return neighbours;
}
string highestScore(list<string> neighbours){
int highest=-1, highestIdx=-1, idx=0, currScore;
if(neighbours.size()==0){
cout<<"Failure: Did not find solution"<<endl;
exit(0);
}
for(string neighbour: neighbours){
currScore=scoreAtNode(neighbour);
if(currScore>highest){
highest=currScore;
highestIdx=idx;
}
idx++;
}
auto it=neighbours.begin();
advance(it,highestIdx);
cout<<"Succesor with highest score:"<<endl;
printSquare(*it);
if(*it==goalState){
cout<<"Reached Goal State"<<endl;
exit(0);
}
visited[*it]=true;
return *it;
}
int main(){
string currNode=inSquare;
cout<<"Start State: "<<endl;
printSquare(currNode);
while(true){
currNode=highestScore(getSuccessors(currNode));
}
}
```

Output :

```
Start State:
_ _ _
|2|8|3|
|1|6|4|
|7|  |5|
_ _ _
Succesor with highest score:
_ _ _
|2|8|3|
|1|  |4|
|7|6|5|
_ _ _
Succesor with highest score:
_ _ _
|2|  |3|
|1|8|4|
|7|6|5|
_ _ _
Succesor with highest score:
_ _ _
|  |2|3|
|1|8|4|
|7|6|5|
_ _ _
Succesor with highest score:
_ _ _
|1|2|3|
|  |8|4|
|7|6|5|
_ _ _
Succesor with highest score:
_ _ _
|1|2|3|
|8|  |4|
|7|6|5|
_ _ _
Reached Goal State
```

Conclusion : C/C++/Java program to solve the above 8-puzzle problem using Steepest Ascent Hill Climbing Algorithm was written and executed successfully.