ARTIFICIAL INTELLIGENCE

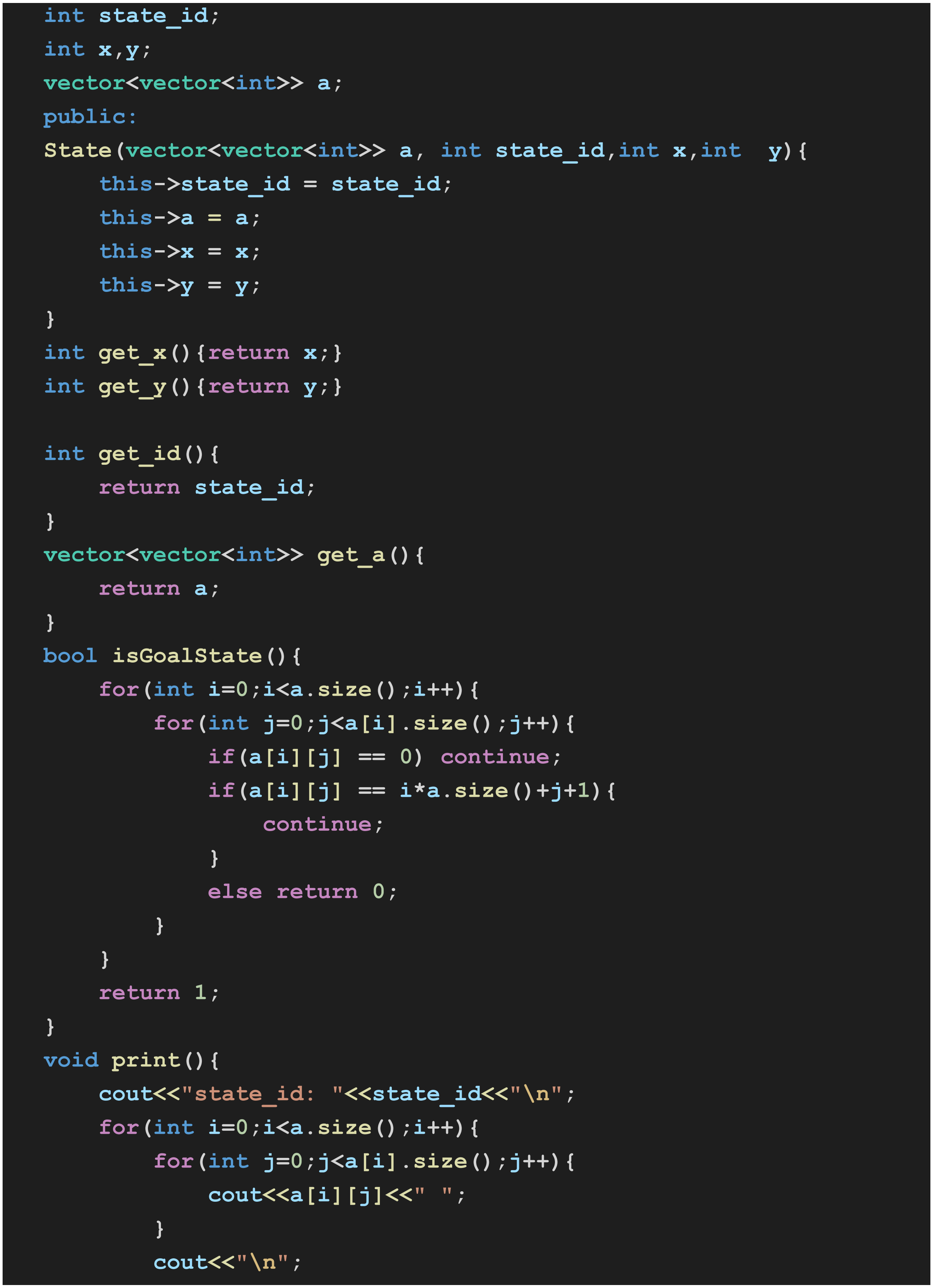
ROLL NO. : 18BCE191
DATE: 18/08/2021

Aim

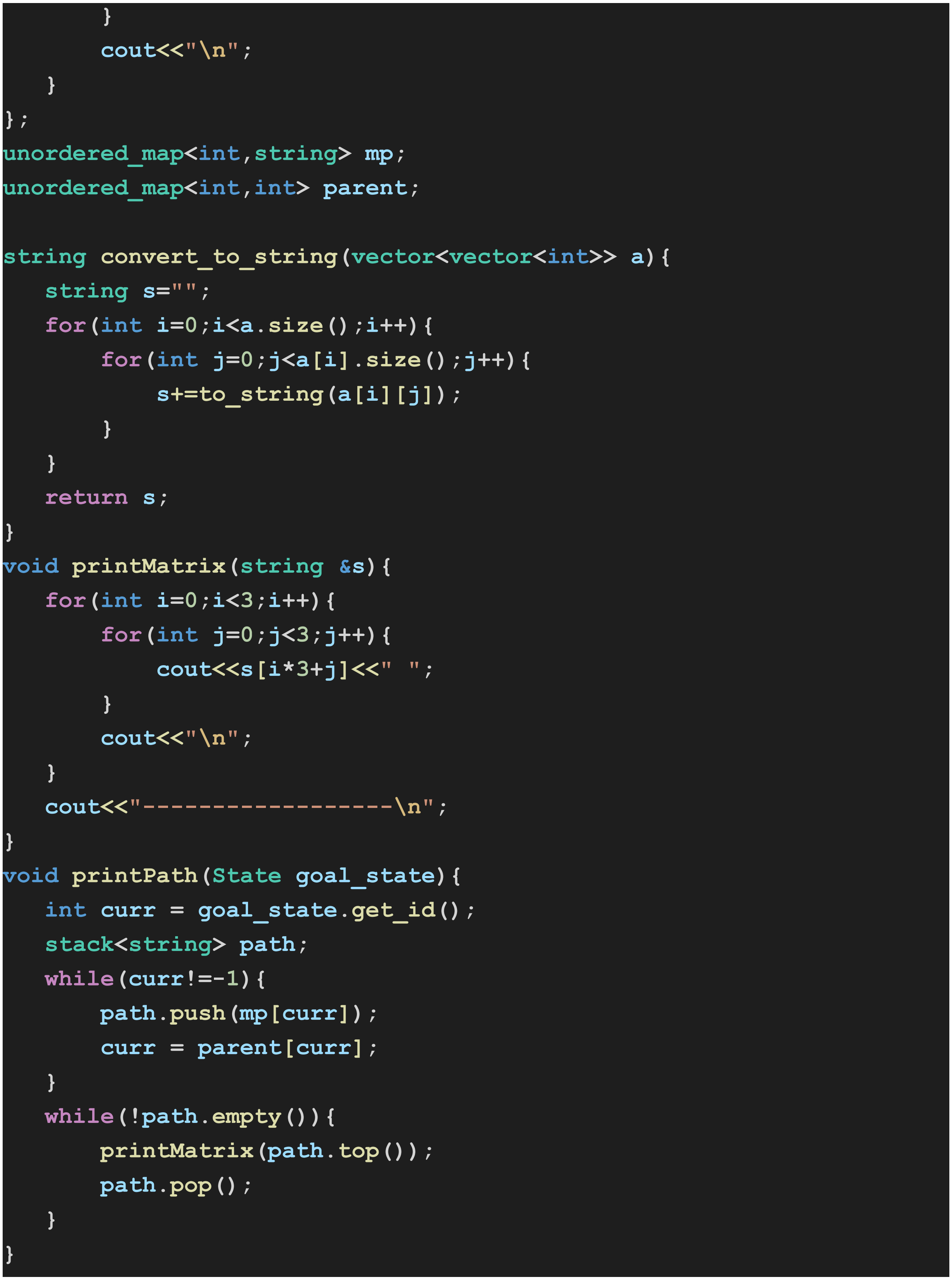
Solve 8-puzzle problem using DFS

Code

```
#include<bits/stdc++.h>
#define FAST ios_base::sync_with_stdio(false);cin.tie();cout.tie();
#define FILE_READ_IN freopen("input2.txt","r",stdin);
#define FILE_READ_OUT freopen("output.txt","w",stdout);
using namespace std;
typedef long long ll;

int dx[4] = {1,0,-1,0};
int dy[4] = {0,1,0,-1};

bool isSolvable(vector<vector<int>>& a){
   int a_flat[9];
   for(int i=0;i<3;i++) for(int j=0;j<3;j++) a_flat[i*3+j]=a[i][j];
   int inversion=0;
   for(int i=0;i<8;i++){
       for(int j=i+1;j<9;j++){
           if(a_flat[i] && a_flat[j] && a_flat[i] > a_flat[j]){
               inversion++;
           }
       }
   }
   return inversion%2==0;
}
class State{
   private:
```

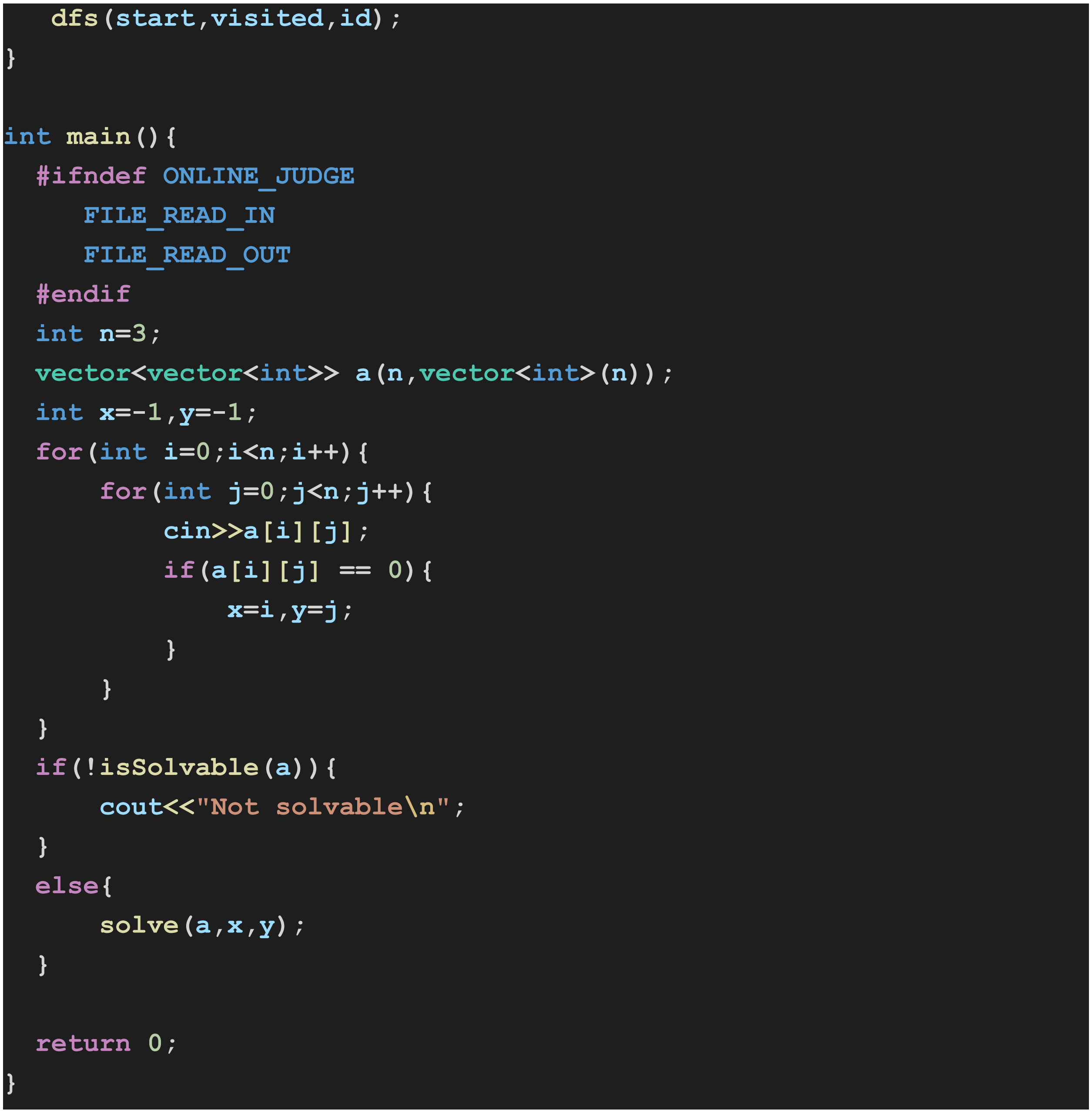


```
int state_id;
int x,y;
vector<vector<int>> a;
public:
State(vector<vector<int>> a, int state_id,int x,int  y){
    this->state_id = state_id;
    this->a = a;
    this->x = x;
    this->y = y;
}
int get_x(){return x;}
int get_y(){return y;}

int get_id(){
    return state_id;
}
vector<vector<int>> get_a(){
    return a;
}
bool isGoalState(){
    for(int i=0;i<a.size();i++){
        for(int j=0;j<a[i].size();j++){
            if(a[i][j] == 0) continue;
            if(a[i][j] == i*a.size()+j+1){
                continue;
            }
            else return 0;
        }
    }
    return 1;
}
void print(){
    cout<<"state_id: "<<state_id<<"\n";
    for(int i=0;i<a.size();i++){
        for(int j=0;j<a[i].size();j++){
            cout<<a[i][j]<<" ";
        }
        cout<<"\n";
```

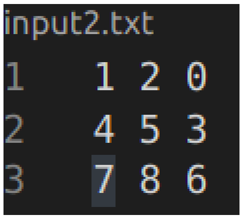
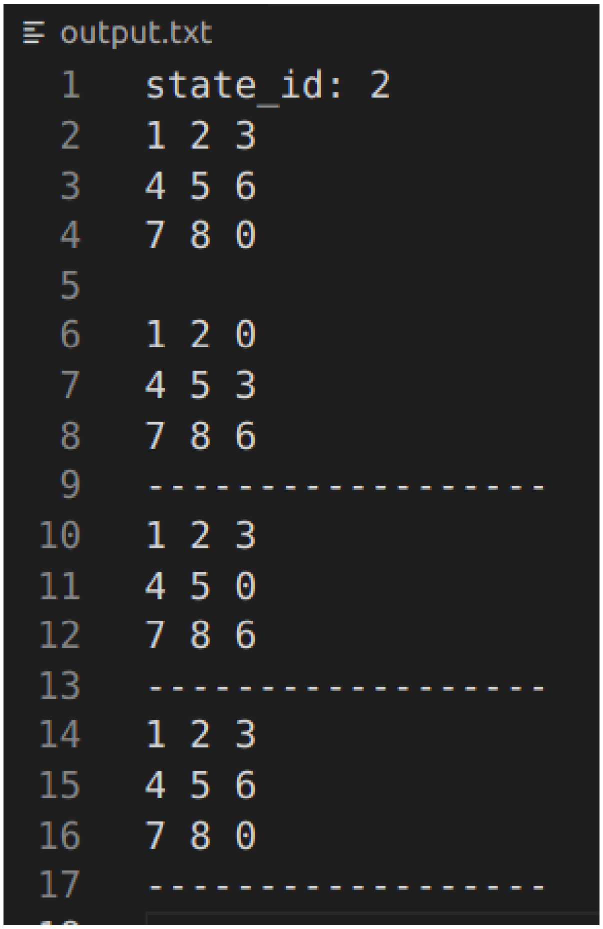
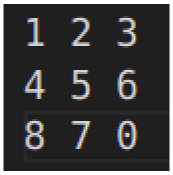
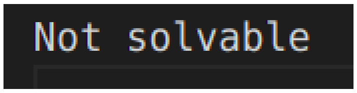


```cpp
        }
        cout<<"\n";
    }
};
unordered_map<int,string> mp;
unordered_map<int,int> parent;

string convert_to_string(vector<vector<int>> a){
    string s="";
    for(int i=0;i<a.size();i++){
        for(int j=0;j<a[i].size();j++){
            s+=to_string(a[i][j]);
        }
    }
    return s;
}
void printMatrix(string &s){
    for(int i=0;i<3;i++){
        for(int j=0;j<3;j++){
            cout<<s[i*3+j]<<" ";
        }
        cout<<"\n";
    }
    cout<<"------------------\n";
}
void printPath(State goal_state){
    int curr = goal_state.get_id();
    stack<string> path;
    while(curr!=-1){
        path.push(mp[curr]);
        curr = parent[curr];
    }
    while(!path.empty()){
        printMatrix(path.top());
        path.pop();
    }
}
```

```cpp
bool dfs(State current_state,set<vector<vector<int>>> & visited,int
&id,int p=-1){
    mp[current_state.get_id()] =
convert_to_string(current_state.get_a());
    parent[current_state.get_id()]=p;

    visited.insert(current_state.get_a());

    if(current_state.isGoalState()){
        current_state.print();
        printPath(current_state);
        return 1;
    }
    for(int i=0;i<4;i++){
        int nx = current_state.get_x()+dx[i];
        int ny = current_state.get_y()+dy[i];
        if(nx<0 || nx>=3 || ny<0 || ny>=3){
            continue;
        }
        vector<vector<int>> n_a = current_state.get_a();

swap(n_a[nx][ny],n_a[current_state.get_x()][current_state.get_y()])
;

        if(visited.count(n_a)) continue;
        State new_state(n_a,++id,nx,ny);

        if(dfs(new_state,visited,id,current_state.get_id())){
            return 1;
        }

    }
    return 0;
}
void solve(vector<vector<int>>& a,int x,int y){
    State start(a,0,x,y);
    set<vector<vector<int>>> visited;
    int id=0;
```



```
   dfs(start,visited,id);
}

int main(){
  #ifndef ONLINE_JUDGE
     FILE_READ_IN
     FILE_READ_OUT
  #endif
  int n=3;
  vector<vector<int>> a(n,vector<int>(n));
  int x=-1,y=-1;
  for(int i=0;i<n;i++){
      for(int j=0;j<n;j++){
          cin>>a[i][j];
          if(a[i][j] == 0){
              x=i,y=j;
          }
      }
  }
  if(!isSolvable(a)){
      cout<<"Not solvable\n";
  }
  else{
      solve(a,x,y);
  }

  return 0;
}
```

Input	**Output**
input2.txt 1 2 0 4 5 3 7 8 6	output.txt state_id: 2 1 2 3 4 5 6 7 8 0 1 2 0 4 5 3 7 8 6 ------------------ 1 2 3 4 5 0 7 8 6 ------------------ 1 2 3 4 5 6 7 8 0 ------------------
1 2 3 4 5 6 8 7 0	Not solvable

** Note: state_id refers to the number of states that are explored to reach the goal state from the initial state.