CS-304-AI -LAB(LAB TASK-1)

ROLL-423135

NAME-HARSHITH SUDA

CODE-1

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

struct Node{
    Node* parent;
    string s;
    int x,y;
    int level;
};

void printVectorOfVector(vector<vector<int>>&v){
    for(int i=0;i<v.size();i++){
        for(int j=0;j<v[0].size();j++){
            // cout<<"value of i:"<<i<<"value of j:"<<j<<" ";
            cout<<v[i][j]<<" ";
        }
        cout<<endl;
    }
    return;
}

pair<vector<vector<int>>,pair<int,int>> randomPuzzleGenerator(){
    srand(time(NULL));
    vector<int>arr={0,1,2,3,4,5,6,7,8};
    for(int i=0;i<100;i++){
        int m=rand()%9;
        int n=rand()%9;
        swap(arr[m],arr[n]);
    }
    vector<int>m1(arr.begin(),arr.begin()+3);
    vector<int>m2(arr.begin()+3,arr.begin()+6);
    vector<int>m3(arr.begin()+6,arr.begin()+9);
    vector<vector<int>>ans={m1,m2,m3};

    int z=0;
    for(int i=0;i<8;i++){
        if(arr[i]==0){
            z=i;
```

```
            // cout<<"value of z:"<<z<<endl;
            break;
        }
    }
    // cout<<"value of z/3:"<<z/3<<endl;
    // cout<<"value of z%3:"<<z%3<<endl;
    return {ans,{z/3,z%3}};
}

void printStringToPuzzle(string s){
    cout<<s[0]<<" "<<s[1]<<" "<<s[2]<<endl;
    cout<<s[3]<<" "<<s[4]<<" "<<s[5]<<endl;
    cout<<s[6]<<" "<<s[7]<<" "<<s[8]<<endl;
    return;
}
string convertPuzzleToString(vector<vector<int>>&puzzle){
    string s;
    for(int i=0;i<3;i++){
        for(int j=0;j<3;j++){
            s+=char(puzzle[i][j]+'0');
        }
    }
    return s;
}
bool isSolvable(vector<vector<int>>&puzzle){
    vector<int>v;
    for(int i=0;i<puzzle.size();i++){
        for(int j=0;j<puzzle[0].size();j++){
            v.push_back(puzzle[i][j]);
        }
    }
    int invcnt=0;
    for(int i=0;i<8;i++){
        for(int j=i+1;j<9;j++){
            if(v[i] && v[j] && v[i]>v[j]){
                invcnt++;
            }
        }
    }
    return (invcnt%2)==0;
}
bool valid(int x,int y,int n){
    if(x>=0 && y>=0 && x<n && y<n){
        return true;
    }
    return false;
}
Node* createNewNode(Node *parentNode,int newX,int newY){
```

```
    Node* node=new Node();
    if(!node){
        return NULL;
    }
    node->parent=parentNode;
    node->x=newX;
    node->y=newY;
    node->level=parentNode->level+1;
    string temp=parentNode->s;
    swap(temp[3*parentNode->x+parentNode->y],temp[3*newX+newY]);
    node->s=temp;
    node->level=parentNode->level+1;
    return node;
}

void backtrackSolution(Node* node){
    vector<string>v;
    while(node){
        v.push_back(node->s);
        node=node->parent;
    }
    reverse(v.begin(),v.end());
    for(int i=0;i<v.size();i++){
        printStringToPuzzle(v[i]);
        cout<<"Next"<<endl;
    }
    return;
}
void BFS8Puzzle(vector<vector<int>>&puzzle,int i,int j,string &goal){
    string temp=convertPuzzleToString(puzzle);
    // temp="431805267";
    Node* node=new Node();
    node->level=0;
    node->parent=NULL;
    node->x=i;node->y=j;
    node->s=temp;
    queue<Node*>q;
    unordered_set<string>s;
    q.push(node);
    int cnt=0;
    vector<int>X={-1,1,0,0};vector<int>Y={0,0,1,-1};
    while(!q.empty()){
        Node* mainNode=q.front();
        q.pop();
        s.insert(mainNode->s);
        cnt++;
        if(mainNode->s==goal){
            backtrackSolution(mainNode);
```

```
            cout<<"Level: "<<mainNode->level<<endl;
            cout<<"Count: "<<cnt<<endl;
            return;
        }
        for(int i=0;i<4;i++){
            if(valid(mainNode->x+X[i],mainNode->y+Y[i],3)){
                string temp1=mainNode->s;
                swap(temp1[3*mainNode->x+mainNode->y],temp1[3*(mainNode-
>x+X[i])+mainNode->y+Y[i]]);
                if(s.find(temp1)==s.end()){
                    Node* tempNode=createNewNode(mainNode,mainNode-
>x+X[i],mainNode->y+Y[i]);
                    q.push(tempNode);
                }
            }
        }
    }
    return;
}


int main(){


    ios_base::sync_with_stdio(0);
    cin.tie(0);

    string goal="123456780";
    pair<vector<vector<int>>,pair<int,int>>puzzle=randomPuzzleGenerator();
    if(isSolvable(puzzle.first)){
        BFS8Puzzle(puzzle.first,puzzle.second.first,puzzle.second.second,goal)
;
    }
    else{
        printVectorOfVector(puzzle.first);
        cout<<"IMPOSSIBLE TO SOLVE";
    }



    // //IMPOSSIBLE CASE
    // vector<vector<int>>p={{0, 7, 6},{1, 8, 3},{2, 5, 4}};
    // if(isSolvable(p)){
    //     BFS8Puzzle(p,0,0,goal);
    // }
    // else{
    //     printVectorOfVector(p);
    //     cout<<"IMPOSSIBLE TO SOLVE";
    // }
```

```
    // SOLVABLE CASE
    // vector<vector<int>>p={{1, 2, 3},{4, 0, 6},{7, 5, 8}};
    // if(isSolvable(p)){
    //     BFS8Puzzle(p,1,1,goal);
    // }
    // else{
    //     printVectorOfVector(p);
    //     cout<<"IMPOSSIBLE TO SOLVE";
    // }


}
```

CODE-2

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

struct Node{
    Node* parent;
    string s;
    int x,y;
    int level;
};


void printVectorOfVector(vector<vector<int>>&v){
    for(int i=0;i<v.size();i++){
        for(int j=0;j<v[0].size();j++){
            cout<<v[i][j]<<" ";
        }
        cout<<endl;
    }
    return;
}


pair<vector<vector<int>>,pair<int,int>> randomPuzzleGenerator(){
    srand(time(NULL));
    vector<int>arr={0,1,2,3,4,5,6,7,8};
    for(int i=0;i<100;i++){
        int m=rand()%9;
        int n=rand()%9;
        swap(arr[m],arr[n]);
    }
```

```
    vector<int>m1(arr.begin(),arr.begin()+3);
    vector<int>m2(arr.begin()+3,arr.begin()+6);
    vector<int>m3(arr.begin()+6,arr.begin()+9);
    vector<vector<int>>ans={m1,m2,m3};

    int z=0;
    for(int i=0;i<8;i++){
        if(arr[i]==0){
            z=i;
            // cout<<"value of z:"<<z<<endl;
            break;
        }
    }
    // cout<<"value of z/3:"<<z/3<<endl;
    // cout<<"value of z%3:"<<z%3<<endl;
    return {ans,{z/3,z%3}};
}

void printStringToPuzzle(string s){
    cout<<s[0]<<" "<<s[1]<<" "<<s[2]<<endl;
    cout<<s[3]<<" "<<s[4]<<" "<<s[5]<<endl;
    cout<<s[6]<<" "<<s[7]<<" "<<s[8]<<endl;
    return;
}
string convertPuzzleToString(vector<vector<int>>&puzzle){
    string s;
    for(int i=0;i<3;i++){
        for(int j=0;j<3;j++){
            s+=char(puzzle[i][j]+'0');
        }
    }
    return s;
}
bool isSolvable(vector<vector<int>>&puzzle){
    vector<int>v;
    for(int i=0;i<puzzle.size();i++){
        for(int j=0;j<puzzle[0].size();j++){
            v.push_back(puzzle[i][j]);
        }
    }
    int invcnt=0;
    for(int i=0;i<8;i++){
        for(int j=i+1;j<9;j++){
            if(v[i] && v[j] && v[i]>v[j]){
                invcnt++;
            }
        }
    }
```

```
    return (invcnt%2)==0;
}
bool valid(int x,int y,int n){
    if(x>=0 && y>=0 && x<n && y<n){
        return true;
    }
    return false;
}
Node* createNewNode(Node *parentNode,int newX,int newY){
    Node* node=new Node();
    if(!node){
        return NULL;
    }
    node->parent=parentNode;
    node->x=newX;
    node->y=newY;
    node->level=parentNode->level+1;
    string temp=parentNode->s;
    swap(temp[3*parentNode->x+parentNode->y],temp[3*newX+newY]);
    node->s=temp;
    node->level=parentNode->level+1;
    return node;
}

void backtrackSolution(Node* node){
    vector<string>v;
    while(node){
        v.push_back(node->s);
        node=node->parent;
    }
    reverse(v.begin(),v.end());
    for(int i=0;i<v.size();i++){
        printStringToPuzzle(v[i]);
        cout<<"Next"<<endl;
    }
    return;
}


void DFS8Puzzle(Node* node,unordered_set<string>&s,string &goal,bool &found){
    if(node->s==goal){
        found=true;
        backtrackSolution(node);
        cout<<"Level: "<<node->level<<endl;
        return;
    }
    vector<int>X={-1,1,0,0};vector<int>Y={0,0,-1,1};
    for(int i=0;i<4;i++){
        if(valid(node->x+X[i],node->y+Y[i],3)){
```

```
            string temp=node->s;
            swap(temp[3*node->x+node->y],temp[(3*(node->x+X[i])+node-
>y+Y[i])]);
            if(s.find(temp)==s.end()){
                Node* tempNode=createNewNode(node,node->x+X[i],node->y+Y[i]);
                s.insert(temp);
                DFS8Puzzle(tempNode,s,goal,found);
                if(found)return;
            }
        }
    }
    return;

}

void DFS8PuzzleDriver(vector<vector<int>>&puzzle,int i,int j,string &goal){
    unordered_set<string>s;
    string temp=convertPuzzleToString(puzzle);
    Node* node=new Node();
    node->s=temp;
    node->x=i;node->y=j;
    node->level=0;
    node->parent=NULL;
    bool found=false;
    s.insert(temp);
    DFS8Puzzle(node,s,goal,found);
    return;
}



int main(){

    ios_base::sync_with_stdio(0);
    cin.tie(0);

    string goal="123456780";
    // pair<vector<vector<int>>,pair<int,int>>puzzle=randomPuzzleGenerator();
    // if(isSolvable(puzzle.first)){
    //     printVectorOfVector(puzzle.first);
    //     DFS8PuzzleDriver(puzzle.first,puzzle.second.first,puzzle.second.sec
ond,goal);
    // }
    // else{
        // printVectorOfVector(puzzle.first);
    //     cout<<"IMPOSSIBLE TO SOLVE";
    // }

    //IMPOSSIBLE CASE
```

```
    vector<vector<int>>p={{0, 7, 6},{1, 8, 3},{2, 5, 4}};
    if(isSolvable(p)){
        DFS8PuzzleDriver(p,0,0,goal);
    }
    else{
        printVectorOfVector(p);
        cout<<"IMPOSSIBLE TO SOLVE";
    }




    //SOLVABLE CASE
    // vector<vector<int>>p={{1, 2, 3},{4, 0, 6},{7, 5, 8}};
    // if(isSolvable(p)){
    //     DFS8PuzzleDriver(p,1,1,goal);
    // }
    // else{
    //     printVectorOfVector(p);
    //     cout<<"IMPOSSIBLE TO SOLVE";
    // }
}
```