|  |
| --- |
| **NAME:** GOKULRAJ V |
| **ROLL\_NO:** 1905014 |

|  |
| --- |
| **DATE:** 27/07/2022 |
| **EXP\_NO:** 01 |

|  |
| --- |
| **MISSIONARIES AND CANNIBALS PROBLEM** |

**Aim**:

To solve the missionaries and cannibals problem.

**Problem Statement:**

In the missionaries and cannibals problem, three missionaries and three cannibals must cross a river using a boat which can carry at most two people, under the constraint that, for both banks, if there are missionaries present on the bank, they cannot be outnumbered by cannibals. The boat cannot cross the river by itself with no people on board.

**DFS CODE:**

#include<bits/stdc++.h>

using namespace std;

//vector {M,C,L/R} 0-L 1-R

void printv(vector<int> v)

{

cout<<v[0]<<" "<<v[1]<<" ";

cout<<" ====== ";

cout<<3 - v[0]<<" "<<3 - v[1]<<" ";

if(v[2]==0)

cout<<" L ";

else

cout<<" R ";

cout<<endl;

}

void printv2(vector<int> v)

{

cout<<v[0]<<" "<<v[1]<<" ";

if(v[2]==0)

cout<<"L";

else

cout<<"R";

cout<<"\n";

}

bool goNxt(vector<int> &state, vector<int> add, set<vector<int>> &hmap)

{

//cout<<"Debug: "<<add[0]<<" "<<add[1]<<endl;

state[0]+=add[0];

state[1]+=add[1];

state[2]= state[2]==1?0:1;

//cout<<"Debug: ";

//printv(state);

if(state[0]<0 || state[0]>3 || state[1]<0 || state[1]>3)

{

state[0]-=add[0];

state[1]-=add[1];

state[2]= state[2]==1?0:1;

return false;

}

//3 3 0 -> m c 2 2 1 -> 3 2 0

if((state[0]>=state[1] || state[0]==0) && (3-state[0] >= 3-state[1] || 3-state[0]==0)) {

if(hmap.count(state)!=0)

{

state[0]-=add[0];

state[1]-=add[1];

state[2]= state[2]==1?0:1;

return false;

}

return true;

}

state[0]-=add[0];

state[1]-=add[1];

state[2]= state[2]==1?0:1;

return false;

}

int main()

{

vector<int> state = {3,3,0};

stack<pair<vector<int>,int>> s;

set<vector<int>> hmap;

vector<vector<int>> nxtStatesL ={{-2,0}, {0,-2}, {-1,0}, {0,-1}, {-1,-1}};

vector<vector<int>> nxtStatesR ={{2,0}, {0,2}, {1,0}, {0,1}, {1,1}};

s.push(make\_pair(state,0));

while(!s.empty())

{

pair<vector<int>,int> tmp = s.top();

//cout<<"ghj\n";

state = tmp.first;

s.pop();

if(hmap.count(tmp.first)!=0)

continue;

hmap.insert(tmp.first);

cout<<tmp.second<<" : ";

printv(tmp.first);

if(tmp.first[0]==0 && tmp.first[1]==0 && tmp.first[2]==1)

break;

if(tmp.first[2]==0)

{

for(auto v:nxtStatesL)

{

if(goNxt(state, v, hmap))

{

s.push(make\_pair(state,tmp.second+1));

state=tmp.first;

}

}

}

else if(tmp.first[2]==1)

{

for(auto v:nxtStatesR)

{

if(goNxt(state, v, hmap))

{

s.push(make\_pair(state,tmp.second+1));

state=tmp.first;

}

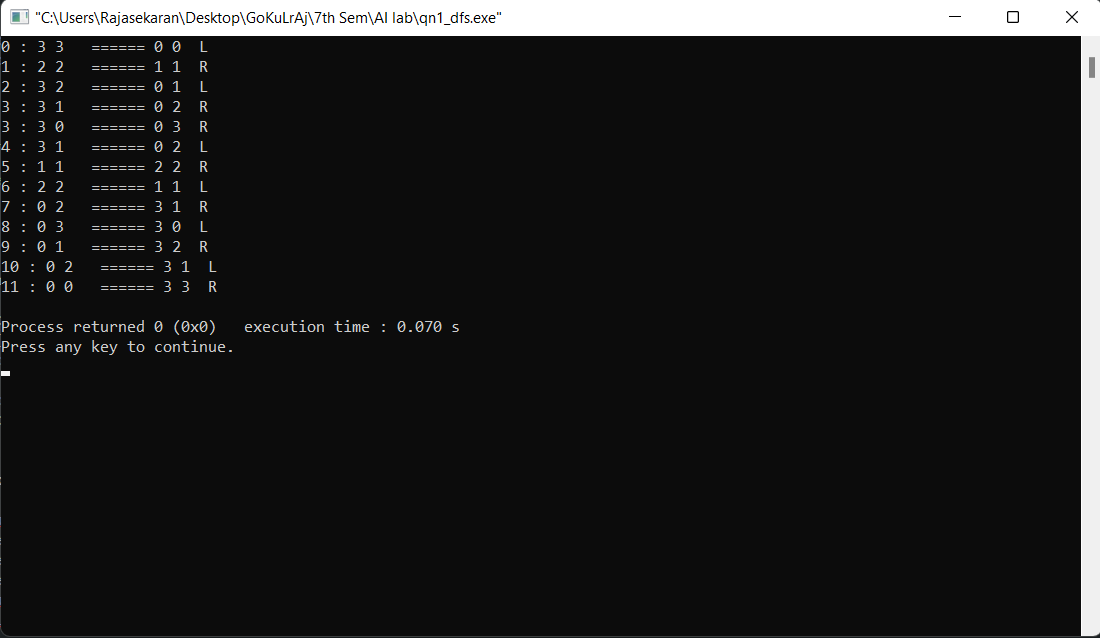
}

}

}

}

**OUTPUT:**



**BFS CODE:**

#include<bits/stdc++.h>

using namespace std;

//vector {M,C,L/R} 0-L 1-R

void printv(vector<int> v)

{

cout<<v[0]<<" "<<v[1]<<" ";

cout<<" ====== ";

cout<<3 - v[0]<<" "<<3 - v[1]<<" ";

if(v[2]==0)

cout<<" L ";

else

cout<<" R ";

cout<<endl;

}

printv2(vector<int> v)

{

if(v[2]==1)

{

cout<<v[0]<<" "<<v[1]<<" L -----> "<<3 - v[0]<<" "<<3 - v[1]<<" R\n";

}

else

{

cout<<3 - v[0]<<" "<<3 - v[1]<<" R -----> "<<v[0]<<" "<<v[1]<<" L\n";

}

}

bool goNxt(vector<int> &state, vector<int> add, set<vector<int>> &hmap)

{

//cout<<"Debug: "<<add[0]<<" "<<add[1]<<endl;

state[0]+=add[0];

state[1]+=add[1];

state[2]= state[2]==1?0:1;

//cout<<"Debug: ";

//printv(state);

if(state[0]<0 || state[0]>3 || state[1]<0 || state[1]>3)

{

state[0]-=add[0];

state[1]-=add[1];

state[2]= state[2]==1?0:1;

return false;

}

//3 3 0 -> m c 2 2 1 -> 3 2 0

if((state[0]>=state[1] || state[0]==0) && (3-state[0] >= 3-state[1] || 3-state[0]==0)) {

if(hmap.count(state)!=0)

{

state[0]-=add[0];

state[1]-=add[1];

state[2]= state[2]==1?0:1;

return false;

}

return true;

}

state[0]-=add[0];

state[1]-=add[1];

state[2]= state[2]==1?0:1;

return false;

}

int main()

{

vector<int> state = {3,3,0};

queue<pair<vector<int>,int>> q;

set<vector<int>> hmap;

vector<vector<int>> nxtStatesL ={{-2,0}, {0,-2}, {-1,0}, {0,-1}, {-1,-1}};

vector<vector<int>> nxtStatesR ={{2,0}, {0,2}, {1,0}, {0,1}, {1,1}};

q.push(make\_pair(state,0));

while(!q.empty())

{

pair<vector<int>,int> tmp = q.front();

state = tmp.first;

q.pop();

if(hmap.count(tmp.first)!=0)

continue;

hmap.insert(tmp.first);

cout<<tmp.second<<" : ";

printv2(tmp.first);

if(tmp.first[0]==0 && tmp.first[1]==0 && tmp.first[2]==1)

break;

if(tmp.first[2]==0)

{

for(auto v:nxtStatesL)

{

if(goNxt(state, v, hmap))

{

q.push(make\_pair(state,tmp.second+1));

state=tmp.first;

}

}

}

else if(tmp.first[2]==1)

{

for(auto v:nxtStatesR)

{

if(goNxt(state, v, hmap))

{

q.push(make\_pair(state,tmp.second+1));

state=tmp.first;

}

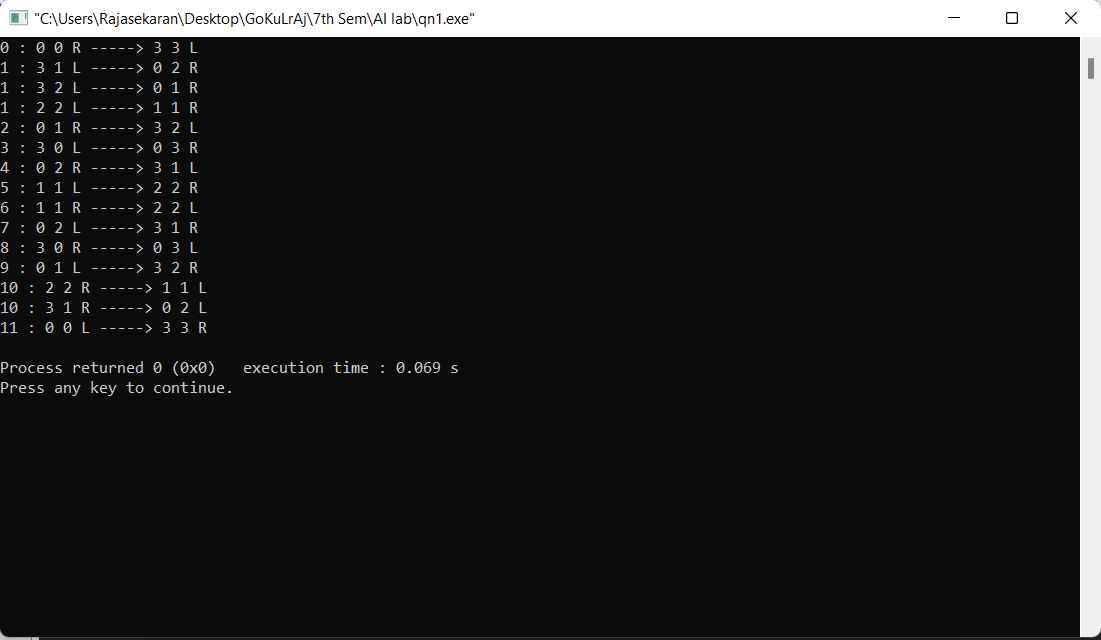
}

}

}

}

**OUTPUT:**



**Result:**

Thus, the code has been successfully executed.