Code :

```
#include <iostream>
#include<list>
#include<map>
#include<string>
using namespace std;

bool DEBUG=false;
list<struct node> queue;
map<string,bool> visited;
int M=3,C=3,B=2;


struct node{
int m;
int c;
char side;
};


bool seen(int m, int n, char side){
// Checks if the node has already been encountered
string temp=to_string(m)+","+to_string(n)+side;
if(visited.count(temp)>0){
return true;
}
visited[temp]=true;
return false;
}
bool moreCannibals(int m, int n){
// Checks if in this state(m,n), either side of the
// river has more cannibals that missionaries
if(m!=0 && m<n){
return true;
}
m=M-m;
n=C-n;
if(m!=0 && m<n){
return true;
}
return false;
}

bool isGoal(int m,int n ,char c){
// Check if goal state is reached
if(m==0 && n==0){
```

```
return true;
}
return false;
}
char sideComplement(char side){
//Returns the complement of current side
if(side=='R')
return 'L';
return 'R';
}

bool isValid(int m, int c,int i,int j, char side){
// Checks if a particular new state is valid
// Is the number of passengers within the capacity of
// the boat and greater than 0?
int numPassengers=0;
if(DEBUG){cout<<i<<" "<<j<<" "<<endl;}
numPassengers=(m-i)+(c-j);
if(numPassengers>B || numPassengers==0){
if(DEBUG){cout<<"not boat"<<endl;}
return false;
}

// more cannibals?
if(moreCannibals(i,j)){
if(DEBUG){cout<<"more cannibals"<<endl;}
return false;
}

//already seen?
if(seen(i,j,sideComplement(side))){
if(DEBUG){cout<<"Already seen"<<endl;}
return false;
}
return true;
}

bool mutate(int m, int c, char side){
// Checks all possible values for number of missionaries
// and cannibals and if the state is valid it pushes it to the
// back of the queue
int i,j;
cout<<"Mutating "<<m<<" "<<c<<" "<<side<<endl;
if(side=='L'){
// Going to the right
```

```
for(i=m;i>=m-B && i>=0;i--){
for(j=c;j>=c-B && j>=0;j--){
if(isValid(m,c,i,j,side)){
// is goal ?
if(isGoal(i,j,sideComplement(side))){
cout<<"Reached goal";
return true;
}
struct node temp={i,j,sideComplement(side)};
cout<<"Pushing "<<i<<" "<<j<<" "<<sideComplement(side)<<endl;
queue.push_back(temp);
}
}
}
}
else{
//Going left
for(i=m;i<=m+B && i<=M;i++){
for(j=c;j<=c+B&& j<=C;j++){
if(isValid(m,c,i,j,side)){
// is goal ?
if(isGoal(i,j,sideComplement(side))){
cout<<"Reached goal"<<endl;
return true;
}
struct node temp={i,j,sideComplement(side)};
cout<<"Pushing "<<i<<" "<<j<<" "<<sideComplement(side)<<endl;
queue.push_back(temp);
}
}
}

}
return false;
}
int main(){
struct node init={M,C,'L'};
queue.push_back(init);

//Add initial node to visited
string temp=to_string(M)+","+to_string(C)+'L';
visited[temp]=true;

bool goal=false;
while(!queue.empty() && !goal){
```

```
struct node s=queue.front();
queue.pop_front();
goal=mutate(s.m,s.c,s.side);
}
}
```

Output :

```
Mutating 3 3 L
Pushing 3 2 R
Pushing 3 1 R
Pushing 2 2 R
Mutating 3 2 R
Mutating 3 1 R
Pushing 3 2 L
Mutating 2 2 R
Mutating 3 2 L
Pushing 3 0 R
Mutating 3 0 R
Pushing 3 1 L
Mutating 3 1 L
Pushing 1 1 R
Mutating 1 1 R
Pushing 2 2 L
Mutating 2 2 L
Pushing 0 2 R
Mutating 0 2 R
Pushing 0 3 L
Mutating 0 3 L
Pushing 0 1 R
Mutating 0 1 R
Pushing 0 2 L
Pushing 1 1 L
Mutating 0 2 L
```

Conclusion : This Experiment was successfully executed and the output was verified.