```
// Project 5

#include <iostream>
#include <limits.h>
#include "d_except.h"
#include <list>
#include <fstream>
#include "d_matrix.h"
#include "graph.h"

using namespace std;

class maze
{
   public:
      maze(ifstream &fin);
      void print(int,int,int,int);
      bool isLegal(int i, int j);

      void setMap(int i, int j, int n);
      int getMap(int i, int j) const;
      void mapMazeToGraph(maze &m, graph &g);

   private:
      int rows; // number of rows in the maze
      int cols; // number of columns in the maze

      matrix<bool> value;
      matrix<int> map;      // Mapping from maze (i,j) values to node
index values
};

void maze::setMap(int i, int j, int n)
// Set mapping from maze cell (i,j) to graph node n.
{
}

int maze ::getMap(int i, int j) const
// Return mapping of maze cell (i,j) in the graph.
{
}

maze::maze(ifstream &fin)
// Initializes a maze by reading values from fin.  Assumes that the
// number of rows and columns indicated in the file are correct.
{
   fin >> rows;
   fin >> cols;

   char x;
```

```
   value.resize(rows,cols);
   for (int i = 0; i <= rows-1; i++)
      for (int j = 0; j <= cols-1; j++)
      {
          fin >> x;
          if (x == 'O')
             value[i][j] = true;
          else
             value[i][j] = false;
      }

   map.resize(rows,cols);
}

void maze::print(int goalI, int goalJ, int currI, int currJ)
// Print out a maze, with the goal and current cells marked on the
// board.
{
   cout << endl;

   if (goalI < 0 || goalI > rows || goalJ < 0 || goalJ > cols)
      throw rangeError("Bad value in maze::print");

   if (currI < 0 || currI > rows || currJ < 0 || currJ > cols)
      throw rangeError("Bad value in maze::print");

   for (int i = 0; i <= rows-1; i++)
   {
      for (int j = 0; j <= cols-1; j++)
      {
          if (i == goalI && j == goalJ)
             cout << "*";
          else
             if (i == currI && j == currJ)
                cout << "+";
             else
                if (value[i][j])
                   cout << " ";
                else
                   cout << "X";
      }
      cout << endl;
   }
   cout << endl;
}

bool maze::isLegal(int i, int j)
// Return the value stored at the (i,j) entry in the maze.
{
```

dest. (rows-1, cols-1)

current vertex v

```
   if (i < 0 || i > rows || j < 0 || j > cols)
      throw rangeError("Bad value in maze::isLegal");

   return value[i][j];
}

void maze::mapMazeToGraph(maze &m, graph &g)
// Create a graph g that represents the legal moves in the maze m.
{
}


int main()
{
   char x;
   ifstream fin;

   // Read the maze from the file.
   string fileName = "maze.txt";

   fin.open(fileName.c_str());
   if (!fin)
   {
      cerr << "Cannot open " << fileName << endl;
      exit(1);
   }

   try
   {

      graph g;
      while (fin && fin.peek() != 'Z')
      {
         maze m(fin);
      }


   }
   catch (indexRangeError &ex)
   {
      cout << ex.what() << endl; exit(1);
   }
   catch (rangeError &ex)
   {
      cout << ex.what() << endl; exit(1);
   }
}
```

(1) Scan maze value
"0" True
g → addNode.
⇒ node id → map.

(2) Scan maze. legal move.
1
↑↓ addEdge(1, 2)
2
addEdge(2, 1)

✓ // empty
// Value.

① m.mapMazeToGraph(&g)
② m.print(dest, start)
③ m.findPathR.(g,)
④ m. . . . NoaR.(g,)