```cpp
#ifndef LIBMAZE_H_INCLUDED
#define LIBMAZE_H_INCLUDED

#include <iostream>
#include <cstdlib>
#include <ctime>
#include <cassert>
#include <sstream>
#include <cctype>

using namespace std;

namespace MazeSpace
{
    typedef int* t_OneDArray;
    typedef int** t_TwoDArray;

    enum enFeatures
    {
        EMPTY,
        WALL,
        ENEMY_UP,
        ENEMY_DOWN
    };

    enum enErrors
    {
        ERR_ARGC = -1,
        ERR_CONV = -2,
        ERR_RANGE = -3
    };

    enum enState
    {
        RUNNING,
        WON,
        LOST,
        QUIT
    };

    const char FEATURES[4] = {'.','|','^','!'};
    const char F_PLAYER = 'P';

    struct stcGame
    {
        t_TwoDArray arrGame;
        int intRows;
        int intCols;
        int intPRow;
        int intPCol;
        enState state;
        bool blnSpotted;
        bool blnSameCol;
        int intMove;
    };

    t_TwoDArray AllocMem(int intRows, int intCols);
    stcGame InitGame(int intRows, int intCols);
    int GetInt(string strNum);
    void PrintWorld(stcGame game);
    void MovePlayer(stcGame& game, char chInput);
    void MoveEnemies(stcGame& game);
    void Dealloc(t_TwoDArray& arrGame, int intRows);
    void Pause();

}

#endif // LIBMAZE_H_INCLUDED

```

```cpp
#include "libMaze.h"

namespace MazeSpace
{
    int GetRand(int intLower, int intUpper)
    {
        int intRange = intUpper-intLower+1;
        return rand()%intRange+intLower;
    }


    void Pause()
    {
        cin.ignore(100,'\n');
        cout << "Press Enter to continue" << endl;
        cin.get();
    }

    t_TwoDArray AllocMem(int intRows, int intCols)
    {
        t_TwoDArray arrGame = new t_OneDArray[intRows];
        for(int r=0;r<intRows;r++)
        {
            arrGame[r] = new int[intCols];
            for(int c=0;c<intCols;c++)
            {
                arrGame[r][c] = EMPTY;
            }
        }
        return arrGame;
    }

    void PlaceFeature(stcGame& game, int intNumFeatures, int intFeature, int intColFrom, int intColTo)
    {
        for(int n=0;n<intNumFeatures;n++)
        {
            int intRow = GetRand(0,game.intRows-1);
            int intCol = GetRand(intColFrom,intColTo);
            while(game.arrGame[intRow][intCol]!=EMPTY)
            {
                intRow = GetRand(0,game.intRows-1);
                intCol = GetRand(intColFrom,intColTo);
            }
            game.arrGame[intRow][intCol] = intFeature;
        }
    }

    void PlaceEnemies(stcGame& game)
    {
        for(int c=1;c<game.intCols-1;c++)
        {
            if(c%2==1)
            {
                int intRow = GetRand(0,game.intRows-1);
                game.arrGame[intRow][c] = GetRand(2,3);
            }
        }
    }

    int GetInt(string strNum)
    {
        stringstream ss {strNum};
        int intNum;
        ss >> intNum;
        if(ss.fail())
        {
            cerr << "Cannot convert string to int" << endl;
            exit(ERR_CONV);
        }
        return intNum;
    }

    stcGame InitGame(int intRows, int intCols)
    {
        t_TwoDArray arrGame = AllocMem(intRows,intCols);
        for(int r=0;r<intRows;r++)
        {
            for(int c=1;c<intCols-1;c++)
```

```
            {
                if(c%2==0)
                    arrGame[r][c] = WALL;
            }
        }
        stcGame game;
        game.arrGame = arrGame;

        //Place the player
        game.intPCol = intCols-1;
        game.intPRow = GetRand(0,intRows-1);
        game.intRows = intRows;
        game.intCols = intCols;
        game.state = RUNNING;
        game.blnSpotted = false;
        game.blnSameCol = false;
        game.intMove = 1;

        //Place the enemies
        PlaceEnemies(game);
        //PlaceFeature(game,intEnemies,ENEMY,1,intCols-2);

        //Open the doors
        for(int c=1;c<intCols-1;c++)
        {
            if(c%2==0)
            {
                int intRow = GetRand(0,intRows-1);
                game.arrGame[intRow][c] = EMPTY;
            }
        }
        return game;
    }

    bool IsInWorld(int intRows, int intCols, int intRow, int intCol)
    {
        return (intRow>=0&&intRow<intRows &&
                intCol>=0&&intCol<intCols);
    }

    void MovePlayer(stcGame& game, char chInput)
    {
        //Reset the move counter if spotted and finished moving
        if(game.blnSpotted && game.intMove==0)
            game.intMove = 2;

        //If not spotted, ensure the movement is normal
        if(!game.blnSpotted)
            game.intMove = 1;

        if(--game.intMove==0)
        {
            int intDRow = game.intPRow;
            int intDCol = game.intPCol;
            switch(chInput)
            {
            case 'w':
                intDRow--;
                break;
            case 's':
                intDRow++;
                break;
            case 'a':
                intDCol--;
                break;
            case 'd':
                intDCol++;
                break;
            }
            if(IsInWorld(game.intRows,game.intCols,intDRow,intDCol))
            {
                //See if the destination contains an enemy
                if(game.arrGame[intDRow][intDCol]>=ENEMY_UP)
                {
                    game.state = LOST;
                    return;
                }

                //See if the existing location contains an enemy
```

```
158                 if(game.arrGame[game.intPRow][game.intPCol]>=ENEMY_UP)
159                 {
160                     game.state = LOST;
161                     return;
162                 }
163 
164                 if(game.arrGame[intDRow][intDCol]!=WALL)
165                 {
166                     game.intPRow = intDRow;
167                     game.intPCol = intDCol;
168                 }
169 
170                 //See if we reached the safe space
171                 if(game.intPCol==0)
172                     game.state = enState::WON;
173             }
174 
175         }
176 
177     }
178 
179     void Dealloc(t_TwoDArray& arrGame, int intRows)
180     {
181         for(int r=0;r<intRows;r++)
182             delete[] arrGame[r];
183         delete[] arrGame;
184         arrGame = nullptr;
185     }
186 
187     void CopyArray(t_TwoDArray arrFrom, t_TwoDArray arrTo, int intRows, int intCols, int
    intExcept)
188     {
189         for(int r=0;r<intRows;r++)
190         {
191             for(int c=0;c<intCols;c++)
192             {
193                 arrTo[r][c] = EMPTY;
194                 if(arrFrom[r][c]<intExcept)
195                     arrTo[r][c] = arrFrom[r][c];
196             }
197         }
198     }
199 
200     void MoveEnemy(stcGame& game, t_TwoDArray arrTemp, int intRow, int intCol, int intFeature)
201     {
202         int intDRow = intRow;
203         int intDCol = intCol;
204         if(intFeature==ENEMY_UP)
205             intDRow--;
206         else
207             intDRow++;
208         if(IsInWorld(game.intRows,game.intCols,intDRow,intDCol))
209         {
210             arrTemp[intDRow][intDCol]=intFeature;
211             arrTemp[intRow][intCol]=EMPTY;
212         }
213         else
214         {
215             if(intFeature==ENEMY_UP)
216                 intFeature=ENEMY_DOWN;
217             else
218                 intFeature=ENEMY_UP;
219             arrTemp[intRow][intCol]=intFeature;
220         }
221 
222         //See if the enemy is in the same column as the player
223         if(intCol==game.intPCol)
224         {
225             game.blnSameCol = true;
226         }
227 
228     }
229 
230     void MoveEnemies(stcGame& game)
231     {
232         //Assume the enemy and the player is not in the same col
233         game.blnSameCol = false;
234         game.blnSpotted = false;
235 
```

```
236           //Move each of the enemies
237           t_TwoDArray arrTemp = AllocMem(game.intRows,game.intCols);
238           CopyArray(game.arrGame,arrTemp,game.intRows,game.intCols,ENEMY_UP);
239           for(int r=0;r<game.intRows;r++)
240           {
241               for(int c=1;c<game.intCols-1;c+=2)
242               {
243                   if(game.arrGame[r][c]>=ENEMY_UP)
244                       MoveEnemy(game, arrTemp, r, c, game.arrGame[r][c]);
245               }
246           }
247           CopyArray(arrTemp,game.arrGame,game.intRows,game.intCols,999);
248           Dealloc(arrTemp,game.intRows);
249
250           //Modify the player movement if the enemy and the player is in the same column.
251           if(game.blnSameCol)
252           {
253               //Find the row the enemy is in
254               int intERow = 0;
255               for(int r=0;r<game.intRows;r++)
256               {
257                   if(game.arrGame[r][game.intPCol]==enFeatures::ENEMY_DOWN ||
      game.arrGame[r][game.intPCol] == enFeatures::ENEMY_UP)
258                       intERow = r;
259
260               }
261
262               int intFeature = game.arrGame[intERow][game.intPCol];
263               if(intFeature==ENEMY_UP)
264               {
265                   if(game.intPRow<intERow)
266                   {
267                       game.blnSpotted=true;
268                       return;
269                   }
270                   game.blnSpotted = false;
271                   return;
272               }
273
274               if(intFeature==ENEMY_DOWN)
275               {
276                   if(game.intPRow>intERow)
277                   {
278                       game.blnSpotted=true;
279                       return;
280                   }
281                   game.blnSpotted = false;
282                   return;
283               }
284           }
285       }
286
287       void PrintWorld(stcGame game)
288       {
289           system("cls");
290           for(int r=0;r<game.intRows;r++)
291           {
292               for(int c=0;c<game.intCols;c++)
293               {
294                   if(r==game.intPRow && c==game.intPCol)
295                       cout << F_PLAYER;
296                   else
297                       cout <<FEATURES[game.arrGame[r][c]];
298                   cout << " ";
299               }
300               cout << endl;
301           }
302           cout << "w: Move Up" << endl
303                << "s: Move Down" << endl
304                << "a: Move Left" << endl
305                << "d: Move Right" << endl
306                << "q: Quit" << endl;
307       }
308
309   }
310
```

```cpp
#include "libMaze.h"
#include <iostream>

using namespace std;
using namespace MazeSpace;

int main(int argc, char** argv)
{
    srand(time(nullptr));
    bool blnContinue = true;
    char chInput = '\0';

    if(argc!=3)
    {
        cerr << "Incorrect num of command line args" << endl;
        exit(ERR_ARGC);
    }

    int intRows = GetInt(argv[1]);
    int intCols = GetInt(argv[2]);
    //Do some range checking here.

    stcGame game = InitGame(intRows,intCols);
    do
    {
        PrintWorld(game);
        cin >> chInput;
        chInput = tolower(chInput);
        switch(chInput)
        {
        case 'w':
        case 's':
        case 'a':
        case 'd':
            MovePlayer(game,chInput);
            break;
        case 'q':
            game.state = QUIT;
            break;
        default:
            cerr << "Please select a valid input" << endl;
            Pause();
        }
        MoveEnemies(game);
        if(game.state!=RUNNING)
            blnContinue = false;
    }while(blnContinue);

    return 0;
}

```