```
/*******************************************************************************
* This program demonstrates a computer version of the puzzle game Rush Hour.
* In this game, the user moves vehicles around the board to allow the red car to make
it to the exit.
* @author Safa Patel
*******************************************************************************/
#include <iostream>
#include <fstream>
#include <vector>
using namespace std;

// Function to display the Rush Hour game board
void displayBoard(char grid[6][6])
{
// Display the game board with cars and the exit
cout << endl;
cout << "--------" << endl;

for (int row = 0; row < 6; row++)
{
for (int column = 0; column < 6; column++)
{
if(column == 0)
{
cout << "|";
}

char a = '-';
if(grid[row][column] == '-')
{
a = '.';
}
else
{
a = grid[row][column];
}
cout << a;

if (row == 2 && column == 5)
{
cout << "=" << endl;
}
```

```
}

if (row != 2)
{
cout << "|" << endl;
}
}

cout << "--------" << endl;
}

// Function to check if a car exists on the board
bool checkCar(char grid[6][6], char car)
{
bool itExists = false;

for(int i = 0; i < 6; i++)
{
for(int j = 0; j < 6; j++)
{
if(grid[i][j] == car)
{
itExists = true;
}
}
}

if (itExists == false)
{
cout << endl;
cout << "That car is not on the board." << endl;
}

return itExists;
}

// Function to check if move is valid
bool isValid(char grid[6][6], char car, char direction)
{
for (int i = 0; i < 6; i++)
{
for (int j = 0; j < 6; j++)
```

```
{
if (grid[i][j] == car)
{
if (direction == 'L' && (j - 1 < 0 || (grid[i][j - 1] != '-' && grid[i][j - 1] !=
car)))
{
return false;
}
else if (direction == 'R' && (j + 1 >= 6 || (grid[i][j + 1] != '-' && grid[i][j + 1]
!= car)))
{
return false;
}
else if (direction == 'U' && (i - 1 < 0 || (grid[i - 1][j] != '-' && grid[i - 1][j] !=
car)))
{
return false;
}
else if (direction == 'D' && (i + 1 >= 6 || (grid[i + 1][j] != '-' && grid[i + 1][j]
!= car)))
{
return false;
}
}
}
}
return true;
}

// Function to move a car on the Rush Hour game board
void moveCar(char grid[6][6], char car, char direction)
{
if (direction == 'L')
{
for (int i = 0; i < 6; i++)
{
for (int j = 0; j < 6; j++)
{
if (grid[i][j] == car)
{
grid[i][j] = '-';
grid[i][j - 1] = car;
```

```
}
}
}
}
else if(direction == 'U')
{
for (int i = 0; i < 6; i++)
{
for (int j = 0; j < 6; j++)
{
if (grid[i][j] == car)
{
grid[i][j] = '-';
grid[i - 1][j] = car;
}
}
}
}

if (direction == 'R')
{
for (int i = 5; i >= 0; i--)
{
for (int j = 5; j >= 0; j--)
{
if (grid[i][j] == car)
{
grid[i][j] = '-';
grid[i][j + 1] = car;
}
}
}
}
else if(direction == 'D' )
{
for (int i = 5; i >= 0; i--)
{
for (int j = 5; j >= 0; j--)
{
if (grid[i][j] == car)
{
grid[i][j] = '-';
```

```
grid[i + 1][j] = car;
}
}
}
}
}

int main()
{
// Initialize variables and open input file
string filename;
cout << "Enter filename: ";
cin >> filename;

ifstream inputFile;
inputFile.open(filename);

// Initialize the Rush Hour game board as a 2D array
char grid[6][6];

if(!inputFile.is_open())
{
cout << "Unable to open the file" << endl;
}

// Read the initial game board configuration from the input file
for (int row = 0; row < 6; row++)
{
for (int column = 0; column < 6; column++)
{
if(inputFile >> grid[row][column])
{

}
else
{
cout << "Error reading from file" << endl;
inputFile.close();
}
}
}
```

```
// Main game loop
while (true)
{
// Display the current game board
displayBoard(grid);

// Prompt the user for their move
string userMove;
cout << "Enter next move (or Q to quit): ";
cin >> userMove;

// Check if the user wants to quit the game
if (userMove.at(0) == 'Q' || userMove.at(0) == 'q')
{
break;
}
else
{
// Extract the car, distance, and direction from the user's input
char car = toupper(userMove.at(0));
int distance = userMove.at(1) - '0';
char direction = toupper(userMove.at(2));

bool itExists = checkCar(grid, car);
if(itExists)
{
// Check if the move is valid and move the car on the game board and check if the
player has won
for(int k = 0; k < distance; k++)
{
if (isValid(grid, car, direction) == true)
{
moveCar(grid, car, direction);
}
}
if (grid[2][5] == 'R' || grid[2][5] == 'r')
{
displayBoard(grid);
cout << "You win! Congratulations!" << endl;
break;
}
}
```

```
}
}
return 0;
}
```