UnluckyRobot

Hi everyone,

The following document describes your first project for this course in detail. The objective of this project is to test the overall programming knowledge that you have acquired throughout this semester so far.

Project Description

X=0, Y=4	X=1, Y=4	X=2, Y=4	X=3, Y=4	X=4, Y=4 End
X=0, Y=3	X=1, Y=3	X=2, Y=3	X=3, Y=3	X=4, Y=3
X=0, Y=2	X=1, Y=2	X=2, Y=2	X=3, Y=2	X=4, Y=2
X=0, Y=1	X=1, Y=1	X=2, Y=1	X=3, Y=1	X=4, Y=1
Start X=0, Y=0	X=1, Y=0	X=2, Y=0	X=3, Y=0	X=4, Y=0 End

A robot walks into a 5*5 grid to collect some points from some of the cells. The objective of the game is to reach one of the end cells **or** to collect above 2,000 points. Of course, like every other game, you will lose if your points fall under a certain range.

Termination Criterion

Starting at the start cell (x=0, y=0), the goal is to keep moving in the grid and collecting points until one of the following conditions is satisfied:

1. The number of the steps/iterations exceeds 20
2. Your total score falls under -1,000
3. Your total score exceeds 2,000
4. The robot reaches one of the end locations (x=4, y=4) **OR** (x=4, y=0).

Rules of Movement

Our robot can only move in one of the following directions: **right(r), left(l), up(u) or down(d)**. It cannot move diagonally.

At the **beginning** of each iteration the void displayInfo function(int x, int y, int itrCount, int totalScore) will be called. It will **print** a message on the screen reporting the current x and y coordinate of the robot, the total score and the number of iterations made so far.

After the void displayInfo function(int x, int y, int itrCount, int totalScore) is launched, the user will be asked to input a character (**'r': right, 'l': left, 'u': up and 'd': down**) to tell the robot where to go next.

1. If the robot moves one cell to the right, its x coordinate is incremented by one and its y coordinate remains unchanged.
2. If the robot moves one cell to the left, its x coordinate is decremented by one and its y coordinate remains unchanged.
3. If the robot moves one cell up, its y coordinate is incremented by one and its x coordinate remains unchanged.
4. If the robot moves one cell down, its y coordinate will be decremented by one and its x coordinate remains unchanged.

List of Variables

There are some variables that you need to define in your main function before entering the control statement. You might want to use more variables, but the following should be present in your code.

List of mandatory variables

1. **int totalScore**: keeps track of the total number of points obtained, initialized as 300
2. **int itrCount**: keeps track of the number of iterations.
3. **int reward**: stores the output of the reward function or the punishOrMercy function in each iteration.
4. **char direction**: stores the direction the user inputs
5. **int x**: stores the x coordinate of the robot, initialized as 0
6. **int y**: stores the y coordinate of the robot, initialized as 0

Rules of Payment

We don't want the game to go on forever. When the robot moves, (in each iteration), it will lose some points according to the following rules:

1. For every cell, the robot moves **right, left, or down** it will lose 50 points.
2. However, if it decides to go **up**, it will only lose 10 points.

The robot should not exit the grid. So, neither the robot's x coordinate nor its y coordinate should be under 0 or above 4 or you will be penalized -2,000 points.

Meaning that at the beginning of each iteration after the printing of the coordinates is done, the user will be asked to input the direction. Then, you need to run the doesExceed function to check if by applying the change the user requested to the x or y coordinate, the new changed coordinate will exceed 4 or if it will fall under 0. If the robot would exceed the grid limits (the doesExceed function returns true), the user will automatically take -2,000 points of damage; the location will remain the same as the previous iteration; the iteration count increases and the user will also have to pay for the movement (-50 or -10 depending on the direction). Otherwise, the location and count will be updated, and the user only has to pay the moving penalty (-50 or -10 depending on the direction).

E.g.: If the robot is at cell (4, 2) and the user enters 'r' they will face a 2,000 point damage (-2,000 added to its total score), the iteration count will be updated; the x coordinate will remain unchanged (4) for the next iteration and the user will also face a -50 damage due to having moved to the right. But if the user inputs 'u', there will be no -2,000 damage and only -10 will be taken out of its total score.

Functions:

void displayInfo function(int x, int y, int itrCount, int totalScore)

This function will be called at the **beginning** of each iteration and it will **print** a message in the console reporting the current x and y coordinate of the robot, the total score, and the number of iterations made so far and moves to the next line(\n).

For the first iteration you will have to print this. You **must** respect the following format for the rest of the iterations as well:

```
For point (X=0, Y=0) at iterations: 0 the total score is: 300
```

boolean doesExceed(int x, int y, char direction)

This function will receive the coordinates of the robot together with the direction the user entered that the robot should move towards. It will return true if the robot would exceed the grid limits after taking a step towards that direction and false otherwise.

int reward()

是说只有在X=4，Y=0或者X=4，Y=0时才有用吗
还是走到第20步的时候

This function will be called when the robot makes a move and ends up in a cell. It returns a number as the reward or the punishment of entering that cell. Once called, the function will roll a dice and return a number based on the following rules:

1. If 1 is displayed it will return -100.
2. If 2 is displayed it will return -200.
3. If 3 is displayed it will return -300.
4. If 4 is displayed it will return 300.
5. If 5 is displayed it will return 400.
6. If 6 is displayed it will return 600.

Back in the main function, the returned number will be stored in the variable **reward**. After that punishOrMercy function will be called**.**

int punishOrMercy(char direction, int reward)

Output: int which will be the value of zero or the initial reward.

reward function中得到负的值并且刚刚选了往上才执行

Functionality: After the result of the reward function is stored, this function should be called. The goal of this function is to help the robot in case it faced a lot of damage in the current step. However, this function will help the robot only if the robot's reward was negative **and** the direction that the user inputted was **up**. This is common practice in Artificial Intelligence (AI) to reward more or penalize less to create incentive to move towards a certain direction, in our case, up.

What does it do (a more detailed explanation): It will first check the sign of the reward value, if it was negative, **then** it will check if the direction is equal to 'u', if that is the case, then it will flip a coin and for zero (tail) it will return zero, and for one (head) it will return the initial reward (the same reward that the function received as an input). 如果是硬币是0的话负的reward就变成0，如果硬币是1就还是要扣分，刚刚reward的负分还要扣

Once back to the main function, the returned value will replace the old reward and be added to your total score.

Once you are out of the control statement (one of the termination criterions has been satisfied), the user will be asked to enter its name. And two methods will be called after that: the toTitleCase() function and the evaluation function will be called. The name input by the user will be converted to title case by using the toTitleCase() function.

String toTitleCase(String str)

This function will bring a string to title case. Assume the input string only contains two words, separated by a space. 让用户输入他的名字

void evaluation(int totalScore)

Functionality: At the end of the game, this function will be called. It takes as input the total score of the robot and just **prints** (does not return anything) a statement based on the value of the total score:

1. If the score is above (>=) 2000 it will print "Victory, your score is xxxxx".
2. Else it will print "Mission failed, your score is xxxxx".

char inputDirection() :

Functionality: The user will be asked to input **'u'** for up, **'d'** for down, **'l'** for left or **'r'** for up. And in case the user enters anything except one of these four letters, it will be kept asked to enter a new char until the entered char matches one of them.

boolean isGameOver(int x, int y, int totalScore, int itrCount):

Functionality: To check if the game is over, the game will be terminated if any of the following condition meets

1. The number of the steps/iterations exceeds 20
2. Your total score falls under -1000
3. Your total score exceeds 2,000
4. You reach one of the end states

Evaluation Form

1. Correctness
2. Clarity
    a. Documentation
    b. Self-documenting names
    c. Indentation
    d. Spaces
    e. No magic numbers (hard-coded number)
3. Efficiency

Restriction

The code you submit has to be 100% your own work. So please refrain from using things that have not been officially taught to you until now in the class such as array, split(), regular expressions, etc.

Result Example

```
For point(X=0, Y=0) at iteration: 0 the total score is: 300
Please input a valid direction: u
Dice: 2, reward: -200
Coin: tail | Mercy, the negative reward is removed.

For point(X=0, Y=1) at iteration: 1 the total score is: 290
Please input a valid direction: r
Dice: 3, reward: -300

For point(X=1, Y=1) at iteration: 2 the total score is: 240
Please input a valid direction: u
Dice: 5, reward: 400

For point(X=1, Y=2) at iteration: 3 the total score is: 630
Please input a valid direction: r
Dice: 4, reward: 300

For point(X=2, Y=2) at iteration: 4 the total score is: 880
Please input a valid direction: u
Dice: 2, reward: -200
Coin: head | No mercy, the negative rewawed is applied.

For point(X=2, Y=3) at iteration: 5 the total score is: 670
Please input a valid direction: r
Dice: 6, reward: 600

For point(X=3, Y=3) at iteration: 6 the total score is: 1220
Please input a valid direction: u
Dice: 2, reward: -200
Coin: tail | Mercy, the negative reward is removed.

For point(X=3, Y=4) at iteration: 7 the total score is: 1210
Please input a valid direction: u
Exceed boundary, -2000 damage applied
Dice: 2, reward: -200
Coin: head | No mercy, the negative rewawed is applied.

For point(X=3, Y=4) at iteration: 8 the total score is: -1000
Please input a valid direction: r
Dice: 6, reward: 600

Please enter your name (only two words): jon snow
Mission failed! Jon Snow, your score is -450
```

Best of luck