Workshop - 9

Workshop Value: 10 marks (4.375% of your final grade)

Please review the following documents:

1. Workshop **Grading Policies**
2. Workshop **Submission Procedures**
3. Workshop **Group Breakdown**

Workshop Overview

It is the post-apocalyptic era, and hardship and depression runs rampant. You are currently at the bottom of a staircase in the basement of an abandoned building looking at what is pure pitch-black darkness ahead (this is the starting point of your journey). You've heard there are some uplifting discoveries to be made in the rooms found in the depths of this building - but this building has a very complex maze of hallways where many who have attempted exploration have not returned. The only way to ensure your safety is to use your flashlight and manage your time carefully so you do not venture to the point of no return!

The flashlight is your lifeline (as is your smarts). Without these things, you will never escape, so it is vital you monitor the flashlight's remaining power to determine how much of the basement you can discover before having to turn around and head back to safety (return to the bottom of the staircase where you started). It is equally vital you keep accurate records of all your movements so you can retrace your steps and return without getting lost!

Your objective is to explore as many rooms as possible and maximize the use of the battery time that remains for your flashlight. The journey is only successful if you can return to the staircase landing before you are forever left in the dark!

Workshop Details

- Your flashlight battery has only 4.5 hours of life
- All hallways are forty (60) steps long
- Each "[***Move***]" unit is 4 steps in distance and takes 1.5 minutes in time
- You must "[***CheckRoom***]" after each "[**Move**]" command until you decide to "[***TurnAround***]"
- You can only travel in the forward direction
- At the end of a hallway, you must make a decision to turn left or right (use "[***ChangeDirection***]")
  Note: This is an endless labyrinth with no known end – each hallway offers two directions at each end!
- You must track your moves to know when you are at the end of a hallway (and when to make a directional change)
- You can only call the "[***TurnAround***]" command **ONCE** (this marks the beginning of your return back to the beginning).
- You can call the "[***TurnAround***]" command at any time but it should only be done when you have maximized the use of your flashlight battery and can return safely to the beginning
- On the return back to the beginning, you must successfully reverse navigate your recorded journey and before the flashlight battery dies.
- You can't "[***CheckRoom***]" to explore rooms when you are on your return journey back to the beginning.

Navigation "Action" Sub-processes

Each "action" describe below is a black box. You must call these black box subprocesses as needed.

[**Move**] Moves forward one unit:
- "Time" should be incremented by 1.5 minutes
- "Distance" should be incremented by 4 steps

[**CheckRoom**] This process should be called after each "Move" action (only when exploring). This process returns "**Room-A**", "**Room-B**", or "**None**" to identify if there is a room to explore or not.

NOTE: If either **Room-A**, or **Room-B** is returned, the user should decide to explore the room or not and if so, call the appropriate room sub-process

[**ChangeDirection**] Rotates your direction 90 degrees (**left** or **right**) in-place without affecting distance or time. This process expects an argument sent to it that identifies "**left**" or "**right**".

[**TurnAround**] Rotates your direction 180 degrees in-place (without affecting distance or time)

Other Important Details…

Each room you discover will take an unknown/variable number of minutes (based on what lies within). Track room exploration time separately from travel/distance time. You will need to develop a tracking method to store each room duration and use this information to calculate predictive estimates in the determination of whether you can risk further exploration time or should turn around and return to safety before your flashlight power runs out.

To help you accomplish these estimates, you should apply the following logic:

1. Maintain an ongoing average room exploration time (recalculate after each room exploration)
2. Keep track of the longest time spent in a room (this will represent the worst-case scenario based on actual data)
3. Apply a pessimistic risk factor in **predicting** the next room exploration time by *adding both the average room exploration time with the room that took the longest time*. This estimated time should be used in the assessment of whether further exploration is possible.
4. You must also factor the time required to return to safety (distance traveled).

Note: When returning, no room explorations can be performed so the time is based purely on the traveled distance (result of a "[***Move***]" command).

ExploreRoom-A

Congratulations you found a room that will provide you with some much-needed fun time playing "X's and O's"! Use the small, dedicated handheld "X's and O's" game machine and enjoy!

X's and O's is a game comprised of a board of 3 squares by 3 squares. This is a 2-player game of which you will be "Player-1" (Player-2 will be the game machine). The rules are simple, each player uses their own identifier ("X" or "O") and places their identifier in one of the empty squares. Turns are alternated until all squares are filled OR when a winner can be declared. The objective is to get 3 matching squares in sequence either horizontally, vertically, or diagonally. The best of 5 games must be played to determine the overall winner.

Black Box processes you can use:

- There are 4 black box processes you can use to simulate the playing of the game.
  - **RandomBinary** – returns a 1 or a 2 which can be used to determine which player should go first.
  - **MakeMove** (Player-n) – automatically assigns a move for the requested Player-n (1 or 2) and places an X or O on the board so that it will block the opponent winning and attempt to win for player n.
  - **CheckWinner** – this will check the board for a winner and return **0** if there is no winner yet, **1** if player 1 wins, **2** if player 2 wins or **3** if there is a tie.
  - **ClearBoard** – removes all X's and O's from the board to prepare for a new game

Constraints

- Do not define the black box processes described above!
- Determine who starts the first game by calling a black box [***RandomBinary***] process that returns a number (1 or 2):
  - **1**: means Player-1 starts
  - **2**: means Player-2 starts
- Prompt the human player (Player-1) to select the unique identifier symbol ("X" or "O") to use. The computer (Player-2) will assume the alternate symbol.
- Each subsequent game's starting player will be whoever won the previous game
- Each player move adds 4 minutes to the explored time
- Each tied game adds another 8 minutes to the explored time
- Each game the human player loses (Player-1), 15 minutes is added to the explored time
- The OUTPUT returned by this process is the total time accumulated to play all three games.

ExploreRoom-B

Congratulations you found another room that will provide you with some much-needed fun time playing "Hang-Person" (formerly known as "Hangman")! Use the small, dedicated handheld "Hang-Person" game machine and enjoy!

This game requires 1 player. The game begins when a random generated hidden secret word (between 10 and 12 characters) is chosen by the game logic. You are shown a series of underscore characters (underlines) that represent the hidden characters of the secret word. You must guess letters one at a time in an attempt to reveal/show as many matching letters in the secret word as you can. If you can correctly guess the word you win otherwise, you lose!

Black Box processes you should use:

- **GenerateWord** – generates a random 10-12 letter word to be guessed.
- **GuessLetter** – the artificial intelligence routine guesses a letter in the word to find a match returning one of the following values:
  - **1**: If it guesses a letter correctly
  - **-1**: If the letter was not contained in the secret word
  - **0**: If it guesses a letter which reveals the word (player automatically wins)
- **GuessWord** – User is prompted for the word to guess, and the AI routine returns:
  - **1**: if the guess matched the secret word
  - **0**: if the guess did not match the secret word

Constraints

- Do not define the black box processes described above!
- A random 5-15 letter word is generated by calling a black box [***RandomWord***] process
- You have a maximum of 8 letter guesses
- Each letter guess that matches a letter in the secret word will add 3 minutes of explore time
- Each letter guess that does not match a letter in the secret word will add another 6 minutes to the explore time
- The logic should permit the player the option to guess the secret word at any time (be sure to prompt the user: Yes or No accordingly).
- If the player does not identify the secret word, 20 minutes is added to the explore time (and the game ends)
- If the player successfully identifies the secret word, they win! Only 2 minutes is added to the explore time (this also applies when a call to "GuessLetter" returns a 0)
- If you run out of letter guesses (have only 8 guesses), you should permit the player to make a final GuessWord. If the outcome is successful, add 5 minutes, otherwise, the player loses, and you must add 25 minutes to the explore time.
- The OUTPUT returned by this process is the total time accumulated to play the game.

Work Breakdown

[Logic 1] Define the main logic for exploring the dungeon (including returning to the starting point). Use the described black box **actions** accordingly and call the "ExploreRoom-A" or "ExploreRoom-B" subprocesses as required when having to explore a room. Don't forget the time analysis summary you should provide the user so they can make an informed decision on when to turn around before the flashlight runs out of power!

[Logic 2] Define the logic for **Room A** which will return the time it took to explore Room A.

[Logic 3] Define the logic for **Room B** which will return the time it took to explore Room B.

Your Task

Individual Logic Assignment

1. Determine your individual assigned logic part based on your member# (see **Group Breakdown** link at the beginning of this document)
2. Where applicable, apply the core components of the **computational thinking** approach to problem solving to help you synthesize a solution
3. Submit your individual assigned part to your professor (see **Submission Procedures** link at the beginning of this document)

Group Solution

1. In the week the workshop is scheduled, you will be working in your assigned sub-group. See **Group Breakdown** link at the beginning of this document for details on how the sub-groups are determined.
2. Please review what is expected as described in the **Grading Policies** link at the beginning of this document.
3. Submit your group solution to your professor (if you are handing in physical paper answers, follow the directions as set by your professor, otherwise, refer to the **Submission Procedures** link at the beginning of this document)

Presentation

Decide among yourselves which member among you in the **sub- group** will be doing a presentation. Priority should be given to those who have not yet done one. Refer to the **Grading Policies**, and **Submission Procedures** links for details on deadlines, expectations and how to submit your work.