CSE 231 Fall 2023

Programming Project 02

This assignment is worth 20 points (2% of the course grade) and must be **completed and turned in before 11:59 PM on Monday, September 11, 2023. After the due date, your score will be deducted by 25% for every 6 hours late or a fraction of it. No submissions will be accepted after 24 hours from the due date.**

Assignment Overview

This assignment will give you more experience on the use of:

1. data types
2. conditionals
3. iteration
4. input/output

The goal of this project is to make calculations necessary for advanced techniques in Super Mario 64.

Assignment Background

Super Mario 64 is a 3D adventure game developed by Nintendo and released in 1996. The main goal of the game is to run around, jump, and collect the various Power Stars scattered around the game's worlds in order to save the princess. Despite this game being made in the '90s, people are still finding new ways to impose challenges on themselves when they play the game. One such way is to do an "A-Button Challenge", where the player attempts to complete the game while minimizing the amount of times they press the "A" button on the controller (this button makes Mario jump).

In order to avoid jumping, however, a variety of different tactics are used. These exploit the way the game is coded in various ways.

The main mechanic we will be looking at is Parallel Universes (PUs). Mario's position is usually the same as the coordinates the game uses to detect whether Mario is standing on a floor or not. However, if Mario travels a huge distance, then these two values will stop being synced up. The variable that holds the floor detection coordinates can hold only much smaller values than the variable that stores Mario's true position. What this creates, in effect, is the idea that there are identical copies of the game world in a grid-like pattern. Each one of these copies is called a Parallel Universe. These can be travelled between if Mario builds up a lot of speed using glitches in the game's code.

The image to the right shows many PUs, in addition to the real universe (circled in red). This is known as the PU grid.

First, begin by watching the YouTube video below. Note that we are not going to code every concept in this video, it is just a nice explanation. The concepts that we are going to code are highlighted below:
https://youtu.be/kpk2tdsPh0A

Parallel Universe Travel (10:30 – 16:02 in the video): Mario's speed is not necessarily the speed the game uses to see if Mario can travel to a PU. The steeper the slope Mario is standing on, the lower his actual, or De Facto speed, is. The game calculates Mario's De Facto speed with the formula:

```
De Facto Speed = Mario's Speed * cos(angle of the slope)
```

Because Mario is doing this travel in one unit of game time, Mario will travel distance units equal to this new speed (that is, distance = speed * time where time = 1). Therefore, the De Facto speed and Mario's distance are the same. In order to travel from one PU to the next, he would need to travel 65,535 units (PU_SIZE). However, the game will divide Mario's Parallel Universe distance into fourths and check that each "quarter step" along the way is valid. So, the easiest way to travel to a PU is to travel one Quadruple Parallel Universe (QPU), which is four PUs. Thus, we need 65,535 * 4 PU distance to travel 1 QPU (65,535 is the length of one PU).
If one of the quarter steps along the way is invalid, Mario will not move to that quarter step. Therefore, it is possible for Mario to travel one to three of the quarter steps, i.e. he doesn't have to only travel all four of the quarter steps.

Scuttlebug Transportation (3:42 – 5:00 in the video): Scuttlebugs are cute little spider enemies that patrol an area around a fixed point called their "home." They won't leave a set distance from this point. However, if Mario touches one of these enemies (which damages Mario and reduces his health points), then the Scuttlebug's home point is changed to be their current location. Mario can then reposition a Scuttlebug however he wants by luring it to a spot, bumping into it, and then continuing to lure it. He can do this until he doesn't have enough health to continue without running out of health points.
For instance, if Mario has 3 health, he could lure the Scuttlebug X units of distance (where X is the radius of patrol), take a hit, and lure it X more units. He then takes another hit and is able to transport the Scuttlebug X more units of distance. He cannot lure it any more however, because he would run out of health points.

Project Description
Your program will make a variety of calculations for the user to make planning a route through a level like is shown in the video above.
Your program must have a `while` loop. Inside of this loop, you will prompt for input. The starter code provided has this `while` loop and prompts. An integer input of 1, 2 or 3 (or "q") is expected at this prompt. Based on the input provided by the user at the prompt, you will complete one of four tasks:

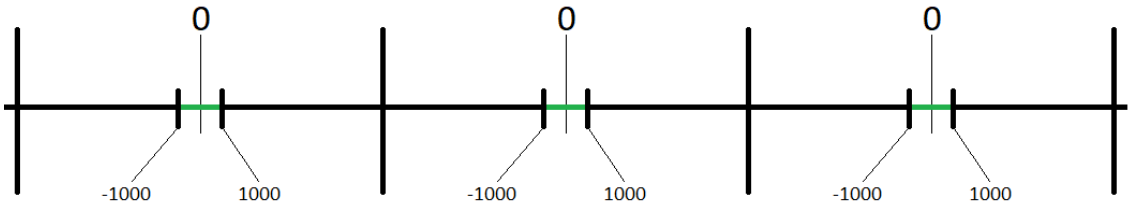
(1): Speed calculator
(2): Parallel Universe navigator
(3): Scuttlebug transportation
(q): Exit the program

1. *Speed calculator*: Prompt for input, asking for how many QPUs they want to travel. Then they will be prompt for the angle of elevation of the slope they are standing on (an angle of 0 means flat ground). Your program will do the appropriate calculations and print out how much speed (not De Facto speed) Mario needs to build up to travel that many QPUs.

a. See the section of the Assignment Background titles “Parallel Universe Travel” for information on how to make the necessary calculations. Knowing the De Facto speed (distance traveled) and the slope, how can you calculate the speed that Mario needs?
b. The number of QPUs is an integer and the slope is float (in radians).
c. You should round off the speed to the nearest integer (use the `round()` function).
d. Remember: distance = speed * time, but time = 1 in our Mario Parallel Universe
e. Hint: Remember that it takes 4 PUs to create one QPU, and the user is telling you how many QPUs they need to travel. Combine these to find the distance travelled (De Facto speed)

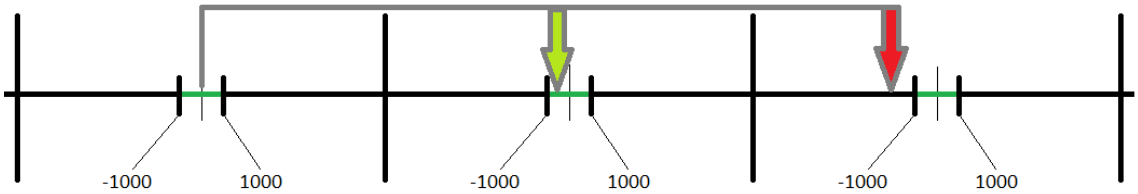
2. *Parallel Universe navigator*: Prompt for input, asking for the angle of elevation of the slope they are standing on (an angle of 0 means flat ground)(this value is a float). Then prompt for how much speed Mario has built up (int). Your program will make the appropriate calculations and print out how many PUs Mario will travel along with his position in this final PU. If an invalid quarter step is found, print out which quarter step was invalid and stop attempting to change Mario’s position.
   a. To keep things simpler, we will assume that Mario is travelling in one of the cardinal directions (North, East, etc.) so you can make this calculation as though Mario is moving through one dimension.

   b. We provide a constant in the starter code called “ISLAND”. This constant tells you how large the ground is inside this PU. So, if ISLAND is 1000 units, then the following graphic demonstrates the pattern for PUs:



   The green part represents the actual level with the black in between being empty space. Note that an ISLAND value of 1000 means that the land extends for 1000 units in both directions, for a total of 2000 units.

   c. Sometimes, Mario’s speed will allow him to make one to three quarter steps before a quarter step fails. When this happens, Mario will only perform the valid quarter steps.



   As you can see in this graphic, Mario’s quarter step of his De Facto speed is less than the size of one PU. So, he makes one quarter step to the second PU, but doesn’t have enough speed to make a valid quarter step to another PU. Because of this, Mario only makes one quarter step. His relative position in this PU will be somewhere between 0 and -1000.

   d. Assume that Mario’s relative position in the universe he begins in is zero.

   e. You will need to calculate Mario’s next position in quarter steps, and if a quarter step is invalid, then that quarter step does not occur and the program should stop. (Hint: you

want a `for` loop that iterates four times, one for each quarter step and `break` out if a quarter-step fails). So, if Mario's De Facto speed was 40,000, you would need to calculate after moving 10,000, 20,000, 30,000, and 40,000 units of distance (remember: the time unit is always 1 so distance always equals speed).

- A quarter step is only valid if:
  - `(0.25 * DeFacto)` is equal to the parallel universe size (`PU_SIZE`) or
  - `abs((0.25*DeFacto + Mario_pos) % PU_SIZE - PU_SIZE) < ISLAND`

- If the quarter step is valid, you should increase the number of parallel universes traveled by Mario by `0.25*DeFacto / PU_SIZE`

- If the quarter step is valid, you change Mario's position:
  - If (`0.25 * DeFacto`) is equal to the parallel universe size (PU_SIZE), then `Mario_pos = 0`
  - Otherwise, make Mario's position `((0.25*DeFacto + Mario_pos) % PU_SIZE) - PU_SIZE`

f. At the end, the program should display the number of `PU` and `Mario_pos` rounded to the nearest integer.

g. See the section of the Assignment Background titled "Parallel Universe Travel" for information on how to make the necessary calculations.

3. *Scuttlebug transportation*: Prompt the user for Mario's current health, known as HP (valid values are 1-8 inclusive). Your program will also prompt for the position of a coin along Mario's route (if the user inputs -1, no coin exists along the route). Print out how much distance a Scuttlebug can be transported with this amount of health (HP). If a coin is within the distance the Scuttlebug can be transported, Mario will heal one health point (increment HP value) and will be able to transport the Scuttlebug farther. For simplicity, we assume that there cannot be more than one coin on the route.
   a. When prompting for HP, check that the value is between 1 and 8 inclusive and reprompt if not. That is, use a `while` loop to keep prompting until a correct value is entered.
   b. Mario can still transport the Scuttlebug by the length of its radius (which is given to you as a constant called "SCUTTLEBUG_RADIUS") if he has at least one health point (HP).
   c. See the section of the Assignment Background titles "Scuttlebug Transportation" for information on how to make the necessary calculations.

4. *Exit*: The program should keep prompting of an input until the user quits and enters `q`.

Assignment Deliverable

The deliverable for this assignment is the following file:

`proj02.py` – the source code for your Python program

Be sure to use the specified file name and to submit it for grading via **Codio** before the project deadline.

Assignment Notes

1. To clarify the project specifications, sample output is appended to the end of this document.
2. Items 1-6 of the Coding Standard will be enforced for this project.
3. We provide a `proj02.py` program for you to start with. It has all the constants (PU_SIZE and ISLAND) and the strings needed to match the output in Codio.
4. You are not allowed to use advanced data structures such as list, dictionaries, classes….
5. Hard coding will result in a score of zero. That is, if you tailor your solution to only work for the test cases instead of working for general cases, you will receive a score of zero.
6. Reminder: collaboration in programming projects is allowed with restrictions: you can discuss with other students in the class only on how to solve the problem. However, when the discussion turns into coding, the discussion needs to stop. Do not show your code to anyone. You can collaborate with GitHub CoPilot only in the coding portion of the project (Note the use of "collaborate". Asking AI to do your work is not collaboration, you will not learn the concepts). See the syllabus for details.

Grading Rubric

```
Computer Project #02                                Scoring Summary
General Requirements:
  ( 5 pts) Coding Standard 1-6
             (descriptive comments, mnemonic identifiers, format, etc...)

Implementation:
 (3 pts) Test Case 1: Speed calculator
 (3 pts) Test Case 2: Parallel Universe navigator
 (3 pts) Test Case 3: Scuttlebug transportation
 (6 pts) Blind Test Case (tests all three options with different input)
```

Test Cases

Test 1

```
Select one of the following options:
    1: Speed calculator
    2: Parallel Universe navigator
    3: Scuttlebug transportation
    q: Exit the program
    Option: 1

How many QPUs do you want to travel? 1

What is the angle of the slope on which Mario is standing? 0

Mario needs 262140 speed

Select one of the following options:
    1: Speed calculator
    2: Parallel Universe navigator
    3: Scuttlebug transportation
    q: Exit the program
    Option: 1

How many QPUs do you want to travel? 3

What is the angle of the slope on which Mario is standing? 0.52

Mario needs 906203 speed

Select one of the following options:
    1: Speed calculator
    2: Parallel Universe navigator
    3: Scuttlebug transportation
    q: Exit the program
    Option: q
```

Test 2

```
Select one of the following options:
    1: Speed calculator
    2: Parallel Universe navigator
    3: Scuttlebug transportation
    q: Exit the program
    Option: 2

What is the angle of the slope on which Mario is standing? 0

What is Mario's speed? 262140

Mario has travelled 4 PUs
Mario's position in this PU: 0

Select one of the following options:
    1: Speed calculator
    2: Parallel Universe navigator
    3: Scuttlebug transportation
    q: Exit the program
```

```
    Option: 2

What is the angle of the slope on which Mario is standing? 0.5236

What is Mario's speed? 604001

Quarter step 4 is invalid!

Mario has travelled 6 PUs
Mario's position in this PU: -900

Select one of the following options:
    1: Speed calculator
    2: Parallel Universe navigator
    3: Scuttlebug transportation
    q: Exit the program
    Option: 2

What is the angle of the slope on which Mario is standing? 0

What is Mario's speed? 362140

Quarter step 1 is invalid!

Mario has travelled 0 PU
Mario's position in this PU: 0

Select one of the following options:
    1: Speed calculator
    2: Parallel Universe navigator
    3: Scuttlebug transportation
    q: Exit the program
    Option: q
```

Test 3

```
Select one of the following options:
    1: Speed calculator
    2: Parallel Universe navigator
    3: Scuttlebug transportation
    q: Exit the program
    Option: 3

What is Mario's current HP? 1

At what distance is the coin placed? Enter -1 if there is no coin. -1

The Scuttlebug can be transported 10 units of distance

Select one of the following options:
    1: Speed calculator
    2: Parallel Universe navigator
    3: Scuttlebug transportation
    q: Exit the program
    Option: 3

What is Mario's current HP? 5

At what distance is the coin placed? Enter -1 if there is no coin. 30
```

```
The Scuttlebug can be transported 60 units of distance

Select one of the following options:
    1: Speed calculator
    2: Parallel Universe navigator
    3: Scuttlebug transportation
    q: Exit the program
    Option: 3

What is Mario's current HP? 8

At what distance is the coin placed? Enter -1 if there is no coin. 9

The Scuttlebug can be transported 80 units of distance

Select one of the following options:
    1: Speed calculator
    2: Parallel Universe navigator
    3: Scuttlebug transportation
    q: Exit the program
    Option: 3

What is Mario's current HP? 9

Invalid amount of HP!

What is Mario's current HP? -1

Invalid amount of HP!

What is Mario's current HP? 8

At what distance is the coin placed? Enter -1 if there is no coin. 120

The Scuttlebug can be transported 80 units of distance

Select one of the following options:
    1: Speed calculator
    2: Parallel Universe navigator
    3: Scuttlebug transportation
    q: Exit the program
    Option: q
```

Test 4

Blind test which tests all three options with different input