15-110: Principles of Computing

Homework 03

Due: 25th August, 2022 at 10:00pm

- You need to complete the Python file for this assignment and submit it to Gradescope.
- There are 100 points.
- You must solve the tasks **individually**, always abiding by the course and CMU's academic integrity policy.

1. (5 points) **Breaking the ice on Piazza**

   This task is easy-peasy. Make a post on Piazza asking a valid question about this homework OR answer one of the questions posted by other students about this homework. Make sure your post is public and follows the academic integrity policy.

2. (15 points) **Back to the Present**

   You got your hands on a time machine that you can use to go back to the past or forward to the future. However, the machine only allows three trips. You can choose to use one, two, or all three trips, and for each trip, you can choose to go to the past or the future. But there is a catch! Each trip is for a fixed number of years, and that cannot be changed.

   For example, if the machine has three trips of 5, 12, and 9 years, you can choose to go to the future for 5 years, and then go to the past for 9 years, ending up 4 years ago. Alternatively, you can choose to take all three trips to the future, ending up 26 years from now.

   You are very attached to the present moment, so although you think the idea of time traveling is interesting, you would like to be able to return to the present time after visiting the past or the future.

   Implement the python function `backToThePresent(t1, t2, t3)` that takes as input the number of years or each trip, and returns `"Yes"` if you can use one, two, or three trips and come back to the present, or `"No"` otherwise.

   Examples:

   - `backToThePresent(22, 5, 22)` returns `"Yes"`
   - `backToThePresent(31, 110, 79)` returns `"Yes"`
   - `backToThePresent(5, 12, 9)` returns `"No"`
   - `backToThePresent(45, 8, 7)` returns `"No"`

3. (20 points) **Getting A**

   If you get more than 90% in a course, an A is (almost always) guaranteed. It is useful to keep track of your grades during the semester to find out if you could still get an A in a course or not.

   Implement the function `canGetA(pts, graded, total)` that takes as input the number of points `pts` you have in the course so far, the number of points `graded` that were already graded, and the

`total` number of points of the course. The function returns `True` if it is still possible for you to get an A, and `False` otherwise.

For example, `canGetA(50,70,100)` should return `False`.

4. (20 points) **Electricity Bill**

   The government of Andrewland is worried about its natural resources. Energy sources are running out faster than expected, and they want to slow down consumption to achieve a sustainable society. One of the suggestions to encourage people to consume less energy is to make those that spend more pay more. The Committee for Mindful Urbanization (CMU) has come up with the following table of prices per kilowatt (kW), depending on how much kWs a building consumed in one month:

kW/month	Price per kW
up to 100	0.25 $
up to 250	0.50 $
up to 500	1.00 $
up to 1000	2.00 $
above 1000	4.00 $

   Implement the python function `electricityBill(kw)` that takes as input the amount of kilowatts a house consumed in one month, and returns its electricity bill.

5. (20 points) **Oldest**

   By looking at two people's birth dates, we can immediately tell who is older. In this task, you will write a python function to do just that!

   Implement the function `oldest(d1, m1, y1, d2, m2, y2)` that takes as input the day, month, and year of birth of person 1 (`d1`, `m1`, and `y1`, respectively), and day, month, and year of birth of person two (`d2`, `m2`, and `y2`, respectively), and returns `1` if person 1 is the oldest, `2` if person 2 is the oldest, and `0` if they are born exactly on the same day.

6. (20 points) **Pomekons Battle**

   After capturing many Pomekons, Maher and Hao decided to go into a battle. The duel rules are simple: each player decides which Pomekon they want to play with, and the Pomekon with the greatest score wins.

   The base score of a Pomekon is calculated using its attack A and defense D strengths according to the following formula:

$$S = \frac{A + D}{2}$$

   In addition to that, if the player's level is an even number, their Pomekons get an additional bonus score.

   Implement the **python function** `winner(b, ha, hd, hl, ma, md, ml)` that takes as input:

   - the bonus (`b`) to be added on even levels;
   - Hao's Pomekon's attack (`ha`) and defense (`hd`) strengths;
   - Hao's level (`hl`)
   - Maher's Pomekon's attack (`ma`) and defense (`md`) strengths; and
   - Maher's level (`ml`).

   The function should return `"Hao"` if Hao wins, `"Maher"` if Maher wins, and `"Draw"` if the duel ends on a draw.