CS 111 Final Exam

INSTRUCTIONS

1. This exam has exactly **3** questions
2. You should submit all of your programs in a single zip file. Submit the zip file through Canvas prior to the end of your exam date.
3. This exam is worth a total of 45 points.
4. The marks for each question are indicated at the beginning of each question.
5. Unlike lab work, there should be no collaboration.
6. If you **do not** complete Question 1, please pledge the exam in a separate text file.

Note

You are writing full programs, this means I expect:

- Appropriate use of functions and especially a main function
- Appropriate docstrings at the top of programs
- Appropriate placements of *import* statements

Questions

1. **Pledge. 15 pts.**

   At Washington and Lee, it is typically required for you to pledge your exams. This exam will be no different, as you will see.

   The pledge is a string of the following form:

   "On my honor, I have neither given nor received any unacknowledged aid on this exam."

   It is typically followed by the string containing your signature on the next line of text. A typed signature will suffice.

   You **will write a program** called *pledge.py*. **This program should do the following, in order.**

   1. **Create a file** called *myPledge.txt* which contains:
      - The text of the pledge
      - Your name on the next line
   2. It then checks the contents of the file; it opens and displays the full text of the file.
   3. It displays the total number of letters in the file. (Only letters, not spaces or punctuation!)

   Note: Please display a full sentence for step 3. For example something like "There are *xyz* letters".

   Include this program, *pledge.py*, in the zip you submit.

2. **Find Index. 15 pts.**

Write a program in a file named *findIndex.py*

This program should contain the following:

1. Write a function named *minIndex* which computes the *index* of the minimum value within a list, beginning the search starting at a specified *startInd* index and ending at a specified *endInd*, excluding the item located at the endInd.

   For example, for a list L=[1, 3, 0, 6, 8, 2, 9, 1], *minIndex(L, 3, 7)* will return the answer *5* because the subsection of L starting at index 3 and ending at index 6 is [6, 8, 2, 9] and L[5] is 2, which is the smallest value.

   Hint: The function needs 3 parameters. The input list, and the two input indices.

2. In the main function of the program, create a list L of 200 random integers (randomly between 0 and 200).

3. Then, in the main function, run the *minIndex* function for every block of 10 numbers in the list L. Starting with the first 10 numbers, then the next 10, and so on.

   For each run, **display** the resulting index and also the corresponding integer in from the list L, on the same line. (Since there are 200 numbers in L in total and you display one result for each block of 10, there should be 20 lines of output displayed in total).

Include *findIndex.py* in the zip you submit.

3. **Dice Game. 15 pts.**

This question involves completing a class file to simulate a dice game. Please study the file *dice_ game.py*. Many methods are already done for you, but some methods (the ones with *pass*) are incomplete. You should complete the methods in this class.

The *main()* function is also already done and should not be changed.

Complete all incomplete methods of the *Player* class.

- The constructor is already done.
- There is one method which rolls a twenty-sided die and returns the result.

Complete all incomplete methods of the *DiceGame* class.

- The constructor creates two *dictionaries*, both of which are keyed by an ID, which is just a number. One dictionary is to store each player, the other dictionary is for storing the number of wins each player has.
- The *resetWins* method should reset the wins which correspond to a player ID to zero.
- The *winCount* method should return the number of wins that a player with the ID has
- The *playGame* method is the most complex, it plays a game with the player (from the game's dictionary) who corresponds to ID1 against the player who corresponds to ID2.
  1. Each player rolls two dice.
  2. The player whose rolls have the larger sum wins
  3. In the case of a tie, the first player wins
  4. Be sure to increase the number of wins for the winner!

Run the program. It should display that "Kunal" (Player ID 421) has roughly 1030 wins.

Include *dice_ game.py* in the zip you submit.