EECE 230C Introduction to Computation and Programming, Sections 1 and 2
Quiz I

March 13, 2018

- The duration of this exam is 2 hours and 55 minutes. Keep in mind that you need around 10 minutes at the end of the duration of the exam to submit your answers. It is your responsibility to make sure your files are correctly submitted.
- The exam consists of 5 problems for 200 points
- You can use all the material in the exam zip file on moodle (lecture slides, source code, programming assignments, and solutions). Once you download the zip file, moodle will be disconnected.
- At the end of the exam, moodle will reopen for exam submission. If you would like to submit your work before the end of the exam, please talk to the proctors for instructions.
- You are asked to submit a single zip file containing your `Python` files (ending with `.py` extension). Failure to do so may lead to a failing grade on the exam. It is your responsibility to make sure your files are correctly submitted.
- You are **NOT** allowed to use the **web**. You are not allowed to use **USB's** or files previously stored on your machine.
- If you get caught violating the above rules or if you communicate with a person other than the exam proctors during the exam, you will immediately get zero and you will be referred to the appropriate disciplinary committee.
- Cell phones and any other unauthorized electronic devices are absolutely not allowed in the exam rooms. They should be turned off and put away.
- The problems are of varying difficulty. Below is a rough ordering estimate of the problems in order of increasing difficulty.
  - Level 0 (20 points): Problem 1
  - Level 1 (80 points): Problems 2.a, 3, 5.a, and nonefficient solution of Problem 4
  - Level 2 (70 points): Problems 2.b, 5.b, and efficient solution of Problem 4
  - Level 3 (30 points): Problem 5.c
- Detailed comments are worth partial credit.
- Plan your time wisely. Do not spend too much time on any one problem. Read through all of them first and attack them in the order that allows you to make the most progress.
- Good luck!

Problem 1 (20 points). Relax

Write a Python script which asks the user to enter two real numbers x and y. Your program is supposed to display their sum $x + y$, their product $x * y$, and their ratio x/y as shown below. If $y = 0$, your program should display an the error message "Can't divide by 0".

Sample input/output:

```
Enter a number x: 5.1

Enter a number y: 3.7
x+y = 8.8
x*y = 18.87
x/y = 1.3783783783783783
```

```
Enter a number x: 5.1

Enter a number y: 0
x+y = 5.1
x*y = 0.0
Can't divide by 0
```

Any correct solution is worth full grade.

Submit your solution in a file called Prob1.py including your name and ID number.

Problem 2 (40 points). Largest power

In this problem, using the `math` module and any other module is not allowed.

a) **(20 points) Largest power of 2 dividing n.** Implement the function `largestPowerOf2Dividing(n)`, which given a positive integer n, finds and returns the largest power of 2 which divides n. That is given n, `largestPowerOf2Dividing(n)` returns the largest i such that 2^i divides n.

Any correct solution is worth full grade.

You don't have to worry about the case when the user input n is not a positive integer.

Test program:

```
print("a)")
print(largestPowerOf2Dividing(1))
print(largestPowerOf2Dividing(2))
print(largestPowerOf2Dividing(3))
print(largestPowerOf2Dividing(4))
print(largestPowerOf2Dividing(5))
print(largestPowerOf2Dividing(20))
print(largestPowerOf2Dividing(48))  # 48 = 3*(2**4)
```

```
a)
0
1
0
2
0
2
4
```

b) **(20 points) Largest power dividing n.** Implement the function `largestPowerDividing(n)`, which given a positive integer n, finds and returns the largest power of an integer greater than 1 which divides n. That is given n, `largestPowerOfDividing(n)` returns the largest i such that x^i divides n, for some integer $x > 1$.

For instance, `largestPowerDividing(50)=2` since $50 = 2 \times 5^2$, and `largestPowerDividing(362063535)=6` since $362063535 = 3 \times 5 \times 17^6$.

Any correct solution is worth full grade.

As in Part (a), you don't have to worry about the case when the user input n is not a positive integer.

Test program:

```
print("b)")                                                    | b)
print(largestPowerDividing(1))                                 | 0 1 (corrected after exam)
print(largestPowerDividing(2))                                 | 1
print(largestPowerDividing(3))                                 | 1
print(largestPowerDividing(4))                                 | 2
print(largestPowerDividing(5))                                 | 1
print(largestPowerDividing(9))                                 | 2
print(largestPowerDividing(25))                                | 2
print(largestPowerDividing(50))                                | 2
print(largestPowerDividing(162))  #162 = 2*(3**4)              | 4
print(largestPowerDividing(362063531))                         | 1
print(largestPowerDividing(362063535)) # 362063535 = 3*5*(17**6) | 6
```

Submit your solution in a file called Prob2.py including your name and ID number.

Problem 3 (20 points). Files

Write a Python script which asks the user to enter a non-negative integer n and creates a new file called `Test`n`.txt` consisting of the following $n+1$ alternating lines :

```
1)EECE 230C is fun
2)EECE 230C is easy
3)EECE 230C is fun
4)EECE 230C is easy
...
```

and so on till the nth line. The $(n+1)$st line should be the string "`---`".

Examples:

- If $n = 0$, your program should create a file called `Test0.txt` whose content is:

  ```
  ---
  ```

- If $n = 1$, your program should create a file called `Test1.txt` whose content is:

  ```
  1)EECE 230C is fun
  ---
  ```

- If $n = 2$, your program should create a file called `Test2.txt` whose content is:

  ```
  1)EECE 230C is fun
  2)EECE 230C is easy
  ---
  ```

- If $n = 23$, your program should create a file called `Test23.txt` whose content is:

  ```
  1)EECE 230C is fun
  2)EECE 230C is easy
  3)EECE 230C is fun
  4)EECE 230C is easy
  5)EECE 230C is fun
  6)EECE 230C is easy
  7)EECE 230C is fun
  8)EECE 230C is easy
  9)EECE 230C is fun
  10)EECE 230C is easy
  11)EECE 230C is fun
  12)EECE 230C is easy
  13)EECE 230C is fun
  14)EECE 230C is easy
  15)EECE 230C is fun
  ```

```
16)EECE 230C is easy
17)EECE 230C is fun
18)EECE 230C is easy
19)EECE 230C is fun
20)EECE 230C is easy
21)EECE 230C is fun
22)EECE 230C is easy
23)EECE 230C is fun
---
```

You don't have to worry about the case when the user input n is negative.

Any correct solution is worth full grade.

Submit your solution in a file called Prob3.py including your name and ID number.

Problem 4 (40 points). Quotient

Recall the definition of quotient and remainder. If x and y are nonnegative integers, dividing x by y results in the quotient q and remainder r, where q and r are nonnegative integer such that $0 \leq r < y$ and $x = q * y + r$.

Examples:

- The quotient of 20 divided by 3 is $q = 6$ $(20 = 6 * 3 + 2)$,
- The quotient of 20 divided by 30 is $q = 0$ $(20 = 0 * 30 + 20)$
- The quotient of 20 divided by 19 is $q = 1$ $(20 = 1 * 19 + 1)$.

In this problem you are asked to compute the quotient without using the modulo operator `%` or the division operators `/` and `//` **except for dividing by** 2. Moreover, using the `math` module and any other module is not allowed; you are asked to solve it using loops.

Implement the function `quotient(x,y)`, which given integers $x \geq 0$ and $y > 0$, returns q.

(Hint: by the definition of the quotient, q is the only integer satisfying: $q * y \leq x$ and $(q+1) * y > x$).

Use `assert` to stop the program if $x < 0$ or $y \leq 0$ and display the error message "Bad input".

Any correct solution is worth 20/40 points. To get full grade, do it efficiently and aim for $O(\log n)$ time. Note that in the last call in the below test program, the nonefficient implementation will practically take forever.

Submit your code in a file called Prob4.py including your name and ID number.

Test program:

```
print(quotient(0,3))
print(quotient(3,3))
print(quotient(20,3))
print(quotient(20,30))
print(quotient(20,19))
print(quotient(320098,33))
print(quotient(3202242409077779409090233 27,339284))
```

```
0
1
6
0
1
9699
943823584099980962583
```

Problem 5 (80 points). Splitting lists into contiguous sublists with equal sums

In this problem, you are allowed to use the built-in `sum(L)` function, which given a list L of numbers, returns the summation of the numbers in L. In principle, you are also allowed to use the slicing operator and list methods, but they are probably not useful, at least not for the efficient solutions of Parts (b) and (c).

a) **(20 points) Check if it can be split into two.** Implement the function `isDivisibleA(L)`, which given a list L of numbers, checks whether or not L can be split into two contiguous sublists of equal sums.

Examples:

- For `L = [1,1]`, `isDivisibleA(L) = True` since `L` can be split into the sublists `[1]` and `[1]`, and the sum of elements in each is 1.
- For `L = [2,1,1]`, `isDivisibleA(L) = True` since `L` can be split into the sublists `[2]` and `[1,1]`, and the sum of elements in each is 2.
- For `L = [2,1,1,10,-6]`, `isDivisibleA(L) = True` since `L` can be split into the sublists `[2,1,1]` and `[10,-6]`, and the sum of elements in each is 4.
- `isDivisibleA(L) = False` for each of the following cases: `L = [3,1,2,3]`, `L = [2,1]`, and `L = [1]`.
- For the empty list `L = []`, `isDivisibleA(L) = True` since `L` can be split into two empty sublists `[]` and `[]`, and the sum of elements in each is 0.

Any correct solution is worth full grade.

Test program:

```
print("a)")
print(isDivisibleA([1,1]))
print(isDivisibleA([2,1,1]))
print(isDivisibleA([2,1,1,10,-6]))
print(isDivisibleA([3,1,2,3]))
print(isDivisibleA([2,1]))
print(isDivisibleA([1]))
```

```
a)
True
True
True
False
False
False
```

b) **(30 points) Check if it can be split into three.** Implement the function `isDivisibleB(L)`, which given a list L of numbers, checks whether or not L can be split into three contiguous sublists of equal sums.

For instance, see the first three lists in the below test program. The equal sum sublists are highlighted by alternating underlines.

To get full grade, do it in $O(n)$ time, where `n = len(L)`.

Test program:

```
print("b)")
print(isDivisibleB([1,1,1]))
print(isDivisibleB([3,1,2,3]))
print(isDivisibleB([2,1,1,10,-6,4]))
print(isDivisibleB([2,1,1,10,-6]))
print(isDivisibleB([2,1,1]))
print(isDivisibleB([1]))
```

```
b)
True
True
True
False
False
False
```

c) **(30 points) Check if it can be split into more than one.** Implement the function `isDivisibleC(L)`, which given a list L of numbers, checks whether or not L can be split into at least two contiguous sublists of equal sums.

For instance, see the first three lists in the below test program. Equal sum sublists are highlighted by alternating underlines.

To get full grade, do it in $O(n^2)$ time, where `n = len(L)`.

Test program:

```
print("c)")
print(isDivisibleC([2,2,4,1,1,2,1,1,1,1,4]))  # 5 sublists
print(isDivisibleC([2,2,4,1,1,2,6,-2]))     # 4 sublists
print(isDivisibleC([2,2,4,8]))  # 2 sublists
print(isDivisibleC([2,2,4,7]))
print(isDivisibleC([1,2]))
print(isDivisibleC([1]))
```

```
c)
True
True
True
False
False
False
```

Submit your code in a file called Prob5.py including your name and ID number.