15-110: Principles of Computing, Spring 2018

Problem Set 3 (PS3)

Due: Friday, February 9 by 2:30PM on Gradescope

HANDIN INSTRUCTIONS

Download a copy of this PDF file. You have two ways to fill in your answers:

1. **Just edit (preferred) - Use any PDF editor (e.g., Preview on Mac, iAnnotate on mobile, Acrobat Pro on pretty much anything) to typeset your answers in the given spaces. You can even draw pictures or take a picture of a drawing and import it in the correct place in the document. That's it. (Acrobat Pro is available on all cluster machines.)**

2. **Print and Scan - Alternatively, print this file, write your answers neatly by hand, and then scan it into a PDF file. This is labor-intensive and must be done by the deadline.**

Once you have prepared your submission, submit it on Gradescope. A link to Gradescope is provided in our course website. DO NOT SUBMIT TO AUTOLAB!

Fill in your answers ONLY in the spaces provided. Any answers entered outside of the spaces provided may not be graded. Do not add additional pages. We will only score answers in the given answer spaces provided. If we cannot read your answer or it contains ambiguous information, you will not receive credit for that answer.

Be sure to enter your full name below along with your section letter (A, B, C, etc.) and your Andrew ID. Submit your work on Gradescope by 2:30PM on the Friday given above.

REMINDER: Sharing your answers with another student who is completing the assignment, even in another semester, is a violation of the academic integrity policies of this course. Please keep these answers to yourself.

Name (First Last) _____

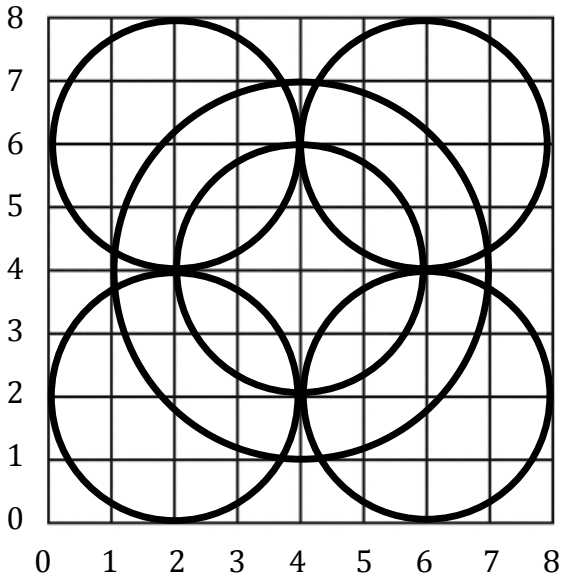
Section _____ Andrew ID _____

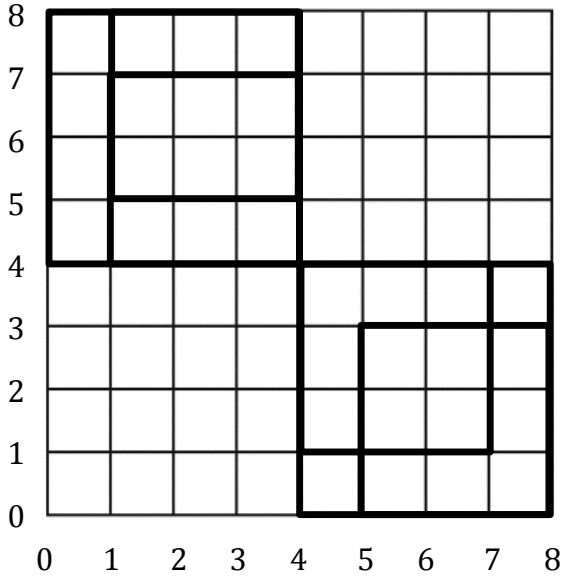
1. (1 pt) Assume you can only give someone a sequence of instructions from the following set of two instructions:

   SQUARE(x,y,s): Draw a square with its top-left corner at (x , y) with a side length of s.

   CIRCLE(x,y,r): Draw a circle centered at (x , y) with a radius of r.

   Write two algorithms made up of SQUARE and/or CIRCLE instructions to draw each of the following pictures, substituting in appropriate values for x, y, s and r in each instruction. Use **as few** instructions as possible to create the pictures.





Answer:

Answer:

2. (2 pts) The Least Common Multiple (LCM) of two numbers x and y is the smallest positive integer that is a multiple of both x and y. (You use this when trying to find the lowest common denominator when adding fractions.)

(a) Consider the following iterative function that computes the LCM of integers x and y, for x > 0 and y > 0:

```
def lcm(x,y):
    p = x * y
    while y != 0:
        temp = y
        y = x % y
        x = temp
        q = p / x
    return q
```

Answer:

x	y	temp	p	q
60	105	---	6300	---

Show how this function computes `lcm(60,105)` by completing the table above that shows the values of each variable at the end of each iteration of the loop. We have started the table for you with the initial values of the variables before the first iteration of the loop. You may not need to use all of the rows of the table.

(b) Consider the following recursive function that computes the LCM of x and y, where x < y and z is a multiple of y (and z is the variable that will eventually hold the answer):

```
def lcm2(x, y, z):
    z = z + y
    if z % x == 0:
        return z
    else:
        return lcm2(x, y, z)
```

To compute the LCM of 60 and 105, we call `lcm2(60,105,0)`. Show how `lcm2(60,105,0)` is computed recursively here by listing the chain of recursive calls that are made until an answer is found. We have started the chain for you below.

Answer: `lcm2(60, 105, 0)`

→ `lcm2(60, 105, 105)`

→

3. (2 pts) Consider the following algorithm to compute the sum of the integers stored in a list named `numlist`:

    1. Set total equal to 0.
    2. Set i equal to 0.
    3. While i is less than the length of numlist, do the following:
        a. Add numlist[i] to total.
        b. Add 1 to i.
    4. Return total.

    (a) Complete the function below that implements the algorithm above in Python using a `while` loop.

```
Answer:     def compute_sum(numlist):
```

    (b) Write a new version of the function that uses a `for` loop instead and does not use list indices (i.e. `numlist[i]` is not allowed).

```
Answer:     def compute_sum2(numlist):
```

4. (3 pts) Recall the implementation of the Sieve of Eratosthenes that we discussed in class:

```
 1   def sift(numlist,k):
 2   # remove all multiples of k from numlist
 3       index = 0
 4       while index < len(numlist):
 5           if numlist[index] % k == 0:
 6               numlist.remove(numlist[index])
 7           else:
 8               index = index + 1
 9       return numlist
10
11   def sieve(n):
12       numlist = []
13       for i in range(2,n+1):
14           numlist.append(i)
15       primelist = []
16       while len(numlist) > 0:
17           primelist.append(numlist[0])
18           lastprime = primelist[len(primelist)-1]
19           numlist = sift(numlist, lastprime)
20       return primelist
```

(a) To return how many primes are less than or equal to n, what single line needs to be changed above, and what do you change it to? (You can test your answer in `python3`.)

Answer:

(b) To return the largest prime less than or equal to n, what single line needs to be changed above, and what do you change it to? (You can test your answer in `python3`.)

Answer:

(c) Suppose we write the `sift` function with the following shorter one that use an iterator:

```
def bad_sift(numlist,k):
# bad sift: tries to remove all multiples of k from numlist
    for value in numlist:
        if value % k == 0:
            numlist.remove(value)
    return numlist
```

See what happens in `python3` when you try to remove all multiples of 2 from the list `[2, 4, 6, 8, 10, 12, 14, 16]`. What result do you get and why does this happen?

Answer (for part c):

(d) A modification we can make to the original sieve function is to remove multiples of 2, 3, ..., up to the square root of n. For example, if we want the primes up to and including 25, we can stop the loop after we remove the multiples of 5, since sqrt(25) is 5. At this point, the rest should be prime. A revised implementation of sieve is given below (assuming we `import math`).

```
11   def sieve(n):
12       numlist = []
13       for i in range(2,n+1):
14           numlist.append(i)
15       primelist = []
16       while numlist[0] <= math.sqrt(n):
17           primelist.append(numlist[0])
18           lastprime = primelist[len(primelist)-1]
19           numlist = sift(numlist, lastprime)
20       return primelist
```

Example for `n = 25` (and `math.sqrt(n) = 5`):

numlist	primelist
[2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25]	[]
[3,5,7,9,11,13,15,17,19,21,23,25]	[2]
[5,7,11,13,17,19,23,25]	[2,3]
[7,11,13,17,19,23]	[2,3,5]

Making the change on line 16 requires another change in the function. Identify which additional line must change and what it must change to so that the returned list has all of the primes up to n in increasing order.

Answer:

5. (2 pts) OPEN LEARNING INITIATIVE (OLI) MODULE – ONLINE ACTIVITY

To help you review iteration, conditionals and lists, there is a module in the OLI system that discusses the key ideas along with exercises interspersed in the readings to see if you understand the core topics.

Go to the OLI system through our Canvas portal.

Read and complete the following lessons in
OLI Module 5: Putting Iterations and Decisions Together :

Euclid's GCD Algorithm (page 32)

Linear Search (page 33)

Keeping state during search (page 34)

Accumulating outputs (page 35)

Note that these will not take you a few minutes. Expect to take several hours to do these lessons since there are exercises to do as you read along. Do not worry if you get the exercises right or wrong. The goal here is to complete them to the best of your ability so we can get a sense of where the class is still having trouble as a group.

To get full credit (2 pts), you must complete all four tasks to the best of your ability. Again, we will not score your answers, just your participation. You will get partial credit (1/2 pt each) for each lesson you complete.

OPTIONAL:
Write down one concept you learned through the OLI module that you didn't know before.