15110 PRINCIPLES OF COMPUTING – LAB EXAM 1 – Fall 2013 A

Name ___________________________ Section _____ Andrew ID ___________________ Machine_____

Directions:

1. ***In your home directory, create a folder named labexam1.***
2. *Write a function in Python for each of the following problems using gedit and store these functions in the labexam1 folder. Test your functions by calling them with python3 -i. Although we give you example/test runs, your function should work on all legal inputs based on the specifications given, and your output should match the examples as closely as possible for full credit. Remember that we will run your code on additional test cases that are not shown on the exam.*
3. *These problems can be done using for loops, while loops, or recursion: your choice (unless otherwise specified).*
4. *Once you are finished, compress the labexam1 folder into a zip file and submit it to AutoLab (http://autolab.cs.cmu.edu) by the end of lab. Do not delete the labexam1 folder from your home directory.*

Below is Python3 syntax reminder for `for` *and* `while` *loops. If we call the functions below with an argument that is a list of numbers they both print the odd items such that each item is printed on a separate line. Note that the* `print` *function can be called with the keyword arguments* `sep` *and* `end`, *defining respectively, the string to be placed between every two printed values and the string to be printed at the end of the print function. For example, using* `print(list[i], end=’’)` *in the examples below would print the values on the same line.*

```
def example1(list):
   for i in range(0,len(list)):
      if list[i]%2 != 0 :
         print(list[i])
```

```
def example2(list):
    i = 0
        while i < len(list):
           if list[i]%2 != 0:
              print(list[i])
           i = i + 1
```

1. (25 pts) Write a Python function `f1(x,y)` (in the file `f1.py` in your `labexam1` folder) that returns the count of numbers that are a multiple of 3 between integers `x` and `y`, inclusive. You can assume that `x` is less than `y`.

Sample usage:

```
>>> f1(4, 19)
5

>>> f1(10, 25)
5
```

2. (25 pts) Write a function `f2(list)` (in the file `f2.py` in your `labexam1` folder) that takes a `list` of integers and returns the value of the smallest integer in the `list`. You may assume that `list` has at least one element. **You may not use the `min` function in your solution.**

Sample usage:

```
>>> f2([4, 2, 5])
2

>>> f2([6, -4])
-4
```

3. (25 pts) Write a function `f3(list)` (in the file `f3.py` in your `labexam1` folder) that takes a list of positive integers and prints a bar graph of X's where the number of X's on each line is given by each integer in the list. At the end of each bar graph, print the length of the bar separated by one space from the bar). You may assume the list has at least one integer. **Your function must use nested loops.**

Sample usage:

```
>>> f3([3, 2, 5, 8, 4])
XXX 3
XX 2
XXXXX 5
XXXXXXXX 8
XXXX 4
```

4. (25 pts) Write a function `f4(list)` (in the file `f4.py` in your `labexam` 1 folder) that stores the steps of a cumulative sum of a list into a new list and returns the list.

You may assume `list` contains at least 1 element. Follow this algorithm:

1. Create a new list *cumulative_sum*.
2. Append *list[0]* to *cumulative_sum.*
3. For each remaining index *i* of *list* do the following:
    a. Add each element at index *i* in *list* to the element at *index-1* in *cumulative_sum* and append the sum to *cumulative_sum.*
4. Return the *cumulative_sum* list.

Hint: If `a` is a list then `a.append(x)` appends the element `x` to the list `a`.

Sample usage:

```
>>> f4([1, 2, 4, 7, 11, 16, 22, 29])
[1, 3, 7, 14, 25, 41, 63, 92]

>>> f4([1, 2, 4, 7, 11, 16, 22, 29, 37, 46, 56, 67, 79, 92])
[1, 3, 7, 14, 25, 41, 63, 92, 129, 175, 231, 298, 377, 469]

>>> f4([1])
[1]
```