CptS355 - Programming Language Design
Spring 2020

Sakire Arslan Ay, PhD

School of Electrical Engineering and Computer Science
Washington State University

Midterm 2 - Sample Exam

- Print your name and WSU ID below.

Name: _____ Sample Solutions _____ WSU ID:_____

Directions: Answer all of the questions. You may have **one 8 1/2 X 11 sheet of notes** during the exam. You may not use a computer, electronic portable device, calculator or phone during the test. Remove all caps and visors.

There are 7 major problems on 7 pages totaling 100 points. **Make sure that you have a complete exam before beginning.**

Write your name on your exam **now**.

Q1 (15pts)	Q2 (21pts)	Q3 (22pts)	Q4 (10pts)	Q5 (20pts)	Q6 (12pts)	Total

1) [15 pts] Directions: Consider the following Python code.

```
x = 0
def add(a):
   global x
   x = a+x
   return x

def grow (c):
   return x+c

def outer (f):
   def inner (g, y):
      return f(y)+g(y)
   return inner
```

For each of the following Python expressions, write the value to which it evaluates, assuming that the expressions are evaluated in a new session (i.e., answers **should not** depend on the previous lines)

a)
```
x=1
L=[1,2,3,4,5]
list(map(grow,L))
```

```
[2, 3, 4, 5, 6]
```

b)
```
x=1
L=[1,2,3,4,5]
list(map(add,L))
```

```
[2, 4, 7, 11, 16]
```

c)
```
x=1
outer(add)(grow,5)
```

```
17
```

d)
```
x=2
outer(add)(grow,x)
```

```
10
```

e)
```
x=2
outer(grow)(add,x)
```

```
8
```

2) [21 pts] Consider the following Python iterator class called `mystery_iter`. Note that the `iterator` is initialized with an iterator object (`iIterable`).

```
class mystery_iter():
   def __init__(self,iIterable):
      self.iterator = iIterable
      self.L = []

   def __next__(self):
      result = None
      for x in self.iterator:
         if x not in self.L:
            self.L.append(x)
            result = x
            break

      if result == None:
         raise StopIteration
      return result

   def __iter__(self):
      return self
```

(a) [6 pts] Explain what sequence the **`mystery_iter`** iterator represents.

The unique values from the iterable input value.

(b) [15 pts] Evaluate the following expressions and write down the output of the print statements.

```
myI = mystery_iter(iter([1,2,1,3,2,1,5,5]))
for i in myI:
    print(i)
```

1 2 3 5

```
myI = mystery_iter(iter(['a','r','s','a','r', 3, 5, 5]))
for i in myI:
    print(i)
```

'a' 'r' 's' 3 5

3) [22 pts]
(a) [10 pts] Define a Python function, `aroundL`, that takes a list of integers as input and returns a list of pairs containing one more and one less than each number in the list. For example, `aroundL ([1,2,3])` should return `[(0,2),(1,3),(2,4)]` as its answer. Your function may involve a loop or can be recursive.

```
def aroundL(L):
    result = []
    for item in L:
        result.append((item-1, item+1))
    return result
```

(b) [6 pts] Re-write the `aroundL` function from problem 4(a) using list comprehension.

```
def aroundL(L):
    return [(item-1, item+1) for item in L ]
```

(c) [6 pts] Re-write the `aroundL` function from problem 4(a) using high order functions (map, reduce, or filter).

```
def aroundL(L):
    return list(map(lambda x: (x-1,x+1),L))
```

4) [10 pts] Define a Python function `wordCount` that is given a list of words (strings) as an argument and returns a dictionary mapping each word to the number of times it appears in the list. Remember to use the dictionary get operation when appropriate to eliminate an if-else.

```
def wordCount(L):
    d = {}
    for word in L:
        d[word] = d.get(word,0)+1
    return d
```

OR

```
def wordCount(L):
    return {word:L.count(word) for word in L}
```

5) **[20 points] (Postscript)**

[10 points] Describe what occurs when each `def`, `dict`, `begin`, `end`, and `mul` operation is executed in the following PostScript program. You must list the operations **in the order they are executed.** For each one say which line number it occurs on and what operands it uses. The first one is done for you as an example. (Of course, the line numbers are not part of the program. There are more blank lines than required answers.)

```
Line
 0        /y 3 def
 1        /f { /z y def  1 dict
 2             begin
 3            /z 4 def
 4             /y 5 def
 5             z y mul
 6            end
 7          } def
 8       f y mul
```

Example:

Line	Operation	Operands
0	def	/y 3
7	def	/f {…}
1	def	/z 3
1	dict	1
2	begin	{}
3	def	/z 4
4	def	/y 5
5	mul	4 5
6	end	
8	mul	3 20

5b) **[5 pts]** What values are on the operand stack when the program finishes execution?

60

5c) **[5 pts]** What is on the dictionary stack when the program finishes execution?

{/y:3, /f:{…}, /z:3}

6) [12 pts] Below is a sequence of four Python assignments. Following those assignments are a number of questions. Answer each question assuming that the 4 assignments have been performed immediately prior to executing the code in that question. The answer to some of the questions may be "an error occurs" in which case describe why it is an error.

```
L = [1, 2, 3]
S = (1, 2, 3)
C = '1 2 3'
D = {3:4, 2:5, 1:6}
```

a) What is L[-1]?

```
3
```

b) What is C[1] (make sure you give an answer of the right type!)?

```
' '       #assume there is space between the numbers in C
```

c) What is S[1:-1]?

```
(2,)
```

d) What is the value of C after executing C[1] = '3'?

```
C[1] = '3' will give an error because Python string are
immutable.
```

e) What is the value of D[3]?

```
4
```