CS100 Fall 2017 Final

December 19, 2017

There are 13 questions on this test. Record your answers to the first 10 questions by circling a letter below. Answer questions 11, 12 and 13 on the attached pages. We have also provided separate scratch pages for writing trial code for the programming problems and tracing code for the multiple choice problems. Work on the scratch pages will not be graded. The value of each question is

1-10 multiple choice (4 points each)
11-13 programming (20 points each)

There is a penalty of one point if your name is not clearly legible and one point if your section is not correct.

Allocate your time accordingly. You may receive partial credit for questions 11, 12 and 13. Answer them as completely as you can. If you finish early, use the extra time to double check your work. When you are done, hand in this answer sheet (including programming question solutions) and the scratch pages and sign the exam attendance sheet.

You may use the summary of Python language elements that is provided. You may not use other notes, books or electronic devices. If you have brought a cell phone or other mobile device you must leave it with the proctor during the exam.

Good luck!

Name (print clearly) ____

Student ID ______________________________ Section (see below) ____________

001 MW 8:30; **003** TR 1:00; **005** TR 10:00; **007** WF 1:00; **009** TF 2:30; **011** TF 4:00; **013** MW 10:00; **015** MW 11:30; **017** TR 8:30; **019** TR 1:00; **021** MW 11:30; **023** TR 11:30; **025** TR 4:00; **101** T 6:00; **103** W 6:00; **105** R 6:00; **107** R 6:00; **109** F 6:00; **H01** MW 1:00; **H03** WF 10:00; **H05** TR 8:30.

Q1 a b c d e

Q2 a b c d e

Q3 a b c d e

Q4 a b c d e

Q5 a b c d e

Q6 a b c d e

Q7 a b c d e

Q8 a b c d e

Q9 a b c d e

Q10 a b c d e

Write code for Questions 11A and 11B here. Use vertical lines for indentation.

Write code for Question 12 here. Use vertical lines for indentation

Write code for Question 13 here. Use vertical lines for indentation.

Multiple choice questions. Four points each.

Question 1

```
def dictTest(d, s):
    isKey = []
    for char in s:
        if char not in d:
            continue
        elif char not in isKey:
            isKey.append(char)
    return len(isKey)

childDictionary = {'a':'apple','b':'bubble'}
print(dictTest(childDictionary, 'sappy'))
```

```
   a) 0
   b) 1
   c) 2
   d) None
   e) none of the above
```

Question 2

```
def isPrefix(a, b):
    prefix = ''
    index = 0
    for letter in b:
        if index >= len(a):
            return a
        elif a[index] != b[index]:
            return prefix
        else:
            prefix += b
            index += 1
    return prefix

word0 = 'asp'
word1 = 'asparagus'
print(isPrefix(word0, word1))
```

```
   a) '' (the empty string)
   b) a
   c) asp
   d) asparagus
   e) none of the above
```

Question 3
```
def isSub(aString):
    words = aString.split()
    left = 0
    right = -1
    for word in words:
        if words[left] not in words[right]:
            return word
        left += 1
        right -= 1
    return ""

s = 'horse sense is nonsense horseradish'
print(isSub(s))
```

```
   a) horse
   b) sense
   c) is
   d) nonsense
   e) none of the above
```

Question 4
```
charSequence = 'toff'
prev = ""
for char in charSequence:
    if prev == "":
        prev = char
        out = ""
        continue
    elif char == prev:
        prev = char
        out = char
        break
    else:
        prev = char
        out = charSequence
print(out)
```

```
   a) "" (the empty string)
   b) t
   c) o
   d) toff
   e) none of the above
```

Question 5
```
digits = {1:'one', 0:'z', 2:'two', 'zero':0}
print(digits[0][1])
```

```
   a) z
   b) e
   c) n
   d) IndexError: string index out of range
   e) none of the above
```

Question 6

```
fred = "Power concedes nothing without a demand"
def property(t, i):
    wordList = t.split()
    d = {}
    for word in wordList:
        length = len(word)
        if length%i == 0:
            continue
        if length not in d:
            d[length] = 1
        else:
            d[length] += 1
    return len(d)

print(property(fred, 2))
```

```
   a) 2
   b) 3
   c) 4
   d) 5
   e) none of the above
```

Question 7

```
fats = ['you', 'were', 'my', 'thrill', 'on', 'Blueberry', 'Hill']
loopCount = 0
idx = 1
while len(fats[idx]) < len(fats):
    idx = len(fats[idx])
    loopCount += 1
print(loopCount)
```

```
   a) 1
   b) 2
   c) 6
   d) 7
   e) none of the above
```

Question 8

```
bools = [True or False, True, False, True and False]
output = 0
for i in range(len(bools)):
    if bools[i]:
        output += 1
    else:
        output *= 2
print(output)
```

```
   a) 3
   b) 4
   c) 6
   d) 8
   e) none of the above
```

Question 9

```
subjects = {'languages':['Python','Java'], 'math':['algebra', 'calculus']}
print(subjects[0][1])
```

```
a) IndexError: string index out of range
b) Java
c) Python
d) calculus
e) none of the above
```

Question 10

The lines below are the content of the file named *lewis.txt*.

Beware the Jabberwock, my son!
The jaws that bite, the claws that catch!
Beware the Jubjub bird, and shun
The frumious Bandersnatch!

After the execution of the following code, what is the content of the file *out.txt*?

```
def lookInto(inFile, outFile, searchString):
    inF = open(inFile, 'r')
    outF = open(outFile, 'w')
    for line in inF:
        instances = line.count(searchString):
        outF.write(str(instances))
    inF.close()
    outF.close()
print(lookInto('lewis.txt', 'out.txt', 'the'))
```

```
a) 1211
b) 11
c) 1110
d) 1 1 1 0
e) none of the above
```

Question 11A (12 points)

Write a **definition line** for a class named State and a **one-line doc string** that describes what a State is. Write definitions for the following methods (functions) in the code for the class:

1. An initialization method. The initialization method should:
   – Take a string parameter, *name*, and assign it as variable *name* of the state being created
   – Create a list variable, *universities*, for the state being created and initialize it to the empty list
2. A method named *addUniversity*. This method should take the name of a university as a string parameter and add it to the list of universities for a that state.
3. A method named *is_home_of*. This method should take the name of a university as a string parameter. If the university is in the state's list of universities it should return True, otherwise it should return False.

Question 11B (8 points)

Assume that the code for the class State (Question 11A) has been saved in a file named *state.py*. Write code that performs the following tasks (each task takes one line):

1. import the module that defines the class State
2. create a state named New Jersey
3. add universities NJIT and Princeton to New Jersey
4. check whether New Jersey is home of MIT and print the result

Question 12 (20 points)

Write a function named wordLineCount with the following input and output:

Input:	a string parameter, *inFile*, that is the name of a file
Output:	return a dictionary in which each unique word in *inFile* is a key and the corresponding value is the number of lines on which that word occurs

The file *inFile* contains only lower case letters and white space.

For example, if the file *ben.txt* contains these lines

tell me and i forget
teach me and i remember
involve me and i learn

then the following would be correct output:

```
>>> print(wordLineCount('ben.txt'))
{'remember': 1, 'and': 3, 'tell': 1, 'me': 3, 'forget': 1, 'learn': 1,
'involve': 1, 'i': 3, 'teach': 1}
```

Question 13 (20 points)

Write a function named *lineStats*. The function *lineStats* takes three parameters:

1. *inFile*, a string that is the name of an input file
2. *outFile*, a string that is the name of an output file
3. *threshold*, an int that is the length above which a word is considered significant

The function *lineStats* should read and analyze each line of the input file and write two statistics, separated by a space, about the line to a corresponding line of the output file.

The two statistics for each line are

1. the number of words
2. the number of distinct significant words (that is, words longer than *threshold*)

Hint: if a word occurs more than once on a line it counts as a single word.

Upper and lower case characters are considered the same ('Word' and 'word' are the same word). The input file contains only upper and lower case letters and white space.

For example, if the file *fish.txt* contains the following lines:

Yes some are red and some are blue
Some are old and some are new

Then the function call

```
lineStats('fish.txt', 'fishOut.txt', 3)
```

should produce an output file *fishOut.txt* with the following content

8 2
7 1