Time In: _____

Time Out: _____

Sign In Number: _____

Week Day: _____

COSC 101C: Introduction to Computing I
Final Exam (Spring 2018)

Honor Code

I agree to comply with the spirit and the rule of the Colgate University Academic Honor Code during this exam and throughout the final exam period. I will not discuss the contents of this exam with other students until the exam period is over at 12pm on Friday, May 11, 2018.

Student Signature: _____

Student Printed Name: _____

Contents

This exam consists of 6 questions totaling 61 points on 15 pages, including this cover sheet and 4 blank scrap pages. Please ensure that you have a complete copy. If you use the scrap pages for work to be graded, indicate this in two places: (1) below the question: indicate which scrap page contains your answer, and (2) on the scrap page: indicate which question you are answering.

Instructions

You have two and a half hours (150 minutes) to complete this exam, and you are responsible for enforcing this time limit on your own. Monitors will write your exam pick-up and return times above. Review the questions so that you can allocate your time properly.

If you want partial credit, show as much of your work and thought process as possible; docstrings and comments can be helpful, but are not required. Remember that you can always define helper functions. Please be sure that your use of indentation is clear for any code you write.

ELECTRONIC DEVICES ARE NOT PERMITTED FOR USE DURING THIS EXAM.
Cell phones, calculators, computers, tablets, and other electronic tools may NOT be used.
Monitors reserve the right to confiscate any electronic devices during this exam.

1. (5 points) Assume that the following statements have already been executed:

```
p = 4
q = 12
r = 3.6
```

For each of the following expressions, evaluate the expression and write the resulting value, or identify the error in the code that would prevent it from running.

(a) `min(p, r)`

(b) `q / round(r)`

(c) `max( int(r), p)`

(d) `q % (p + r // 2)`

(e) `r < q / p`

2. (5 points) Assume that the following statements have already been executed:

```
a = 'University'
b = { a : 'Colgate', 'b' : 'Hamilton', 13 : 'Oak Drive' }
c = [ a[1], a[-1], a.split('i'), a ]
```

For each of the following expressions, evaluate the expression and write the resulting value, or identify the error in the code that would prevent it from running.

(a) `b['b'][-1] in a`

(b) `b['a'][3:]`

(c) `c[2] + [a]`

(d) `len(c) - len(b)`

(e) `a[:3] + a[-2] + a[-1]`

3. (16 points total) This question has three parts, each asking you to trace through code.

   (a) (4 points) What is the output of the following program?

```
def func(lst):
    newlst = lst[:]
    val = 0
    num = len(lst)
    while val < num:
        newlst[val] += val
        val += newlst[val]
    return newlst

alst = func([2,1,4,8,3,1])
print(alst)
```

   (b) (4 points) What is the output of the following program?

```
def funcB(a,b):
    newa = a.copy()
    i = 0
    while i < len(b):
        if b[i] in newa.keys():
            b.append(newa[i])
        else:
            newa[b[i]] = i
        i += 1
    return newa

var1 = {1:3,0:2}
var2 = [2,1]
retval = funcB(var1, var2)
print(var1)
print(var2)
print(retval)
```

(c) (4 points) What is the output of the following program?

```
def funcC(string):
    if string == '':
        return ''
    else:
        return string[-1] * (len(string) % 2 + 1) + funcC(string[:-1])

print(funcC('earth'))
```

4. (14 points total) This question has three parts based on the following function.

```
def flip(lst):
    n = []
    for i in lst:
        if i % 2 == 0:
            n = n + [i]
        else:
            n = [i] + n
    return n

print(flip([4, 1, 7, 2, 5]))
```

(a) (4 points) What does the above program output?

(b) (5 points) Write a function `flip_while` that produces the same output as `flip` using a **`while`** loop instead of a **`for`** loop.

(question continues on next page)

(question continued from previous page)

(c) (5 points) Write a function `flip_recursive` that produces the same output as `flip` using recursion; no loops allowed.

5. (10 points) Write a function `longest_synonyms(d, lst)` that takes a dictionary `d` and a list of words `lst` as parameters. You should assume the values of the dictionary are lists of synonyms (i.e., words with the same meaning). The function should return a list containing the longest synonym for each word in `lst`. If there is a tie, then any of the longest synonyms may be chosen.

   For example:

```
>>> words = { 'happy': ['cheerful', 'delighted', 'glad'],
              'relaxed': ['calm', 'serene', 'tranquil'],
              'sunny': ['bright', 'clear'], 'warm': ['pleasant'] }
>>> longest_synonyms(words, ['happy', 'warm', 'sunny'])
['delighted', 'pleasant', 'bright']
```

6. (15 points total) This question has two parts and is continued on the following pages.

   According to the birthday paradox, there is a 50% chance that in a random group of 23 people two people will have the same birthday. Your task is to write a program that takes the birthdays of all people in a group and determines whether at least two people in the group have the same birthday.

   The birthdays are stored in a file. Each line of the file contains a name, a month, and a date, separated by spaces.

   For example, a short file `presidents.txt` contains the following:

```
Washington February 22
Polk November 2
Lincoln February 12
Harding November 2
Obama August 4
Trump June 14
```

   According to this file, Polk and Harding have the same birthday.

   Your program will be written as multiple functions, described on the following pages.

(question continues on next page)

(question continued from previous page)

(a) (8 points) Write a function `read_birthdays` that takes as a parameter the name of a file of birthdays. This function will return a dictionary with dates as the keys and a list of people born on the corresponding day as the values. For example:

```
>>> read_birthdays('presidents.txt')
{'February 22': ['Washington'], 'November 2': ['Polk', 'Harding'],
'February 12': ['Lincoln'], 'August 4': ['Obama'], 'June 14': ['Trump']}
```

Your function should return `None` if the file is not found. (Hint: Python raises a `FileNotFound` error if it is unable to open a file.) You can assume that opened files will be in the format described on the previous page.

(question continues on next page)

(question continued from previous page)

(b) (7 points) Write a function `find_duplicates` that takes as a parameter a dictionary with dates as the keys and a list of people born on the corresponding day as the values. This function will return a list of strings, where each string contains a date and the names of two or more people who have that same birthday. For example:

```
>>> bdays = {'February 22': ['Washington'], 'November 2': ['Polk', 'Harding'],
'February 12': ['Lincoln'], 'August 4': ['Obama'], 'June 14': ['Trump']}
>>> find_duplicates(bdays)
['November 2: Polk and Harding']
```

You are required to use the `read_birthdays` function from part (a). You can assume the function works as described (regardless of whether your answer is correct or not).

(This page is intentionally blank. Label any work with the corresponding problem number.)

(This page is intentionally blank. Label any work with the corresponding problem number.)

(This page is intentionally blank. Label any work with the corresponding problem number.)

(This page is intentionally blank. Label any work with the corresponding problem number.)