

# COSC 101, Exam #2

## 12 April 2018

Name: \_\_\_\_\_ Section: MWF 9:20 / TR 8:30 / TR 9:55

Instructions and advice:

- Do not open the exam until instructed to do so.
- Write your name and circle your section time.
- You have 90 minutes to complete this exam; use your time wisely.
- There are 5 questions and a total of 50 points available for this exam. Don't spend too much time on any one question.
- If you want partial credit, show as much of your work and thought process as possible.
- Since indentation is important in Python, please be sure that your use of indentation is obvious for any code you write.
- When defining functions, it is not necessary to write docstrings nor is it necessary to write comments.
- If you run out of space while answering a question, you can continue your answer on one of the scrap pages at the end of the exam. If you do so, be sure to 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.

Question	Points	Score
1	8	
2	12	
3	8	
4	10	
5	12	
Total:	50	

1. (8 points) Assume that the following statements have already been executed:

```
a = 'April'
b = 12
c = [b, a, 2018]
d = [4, [3, 2], 1]
```

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) `a[:3]`

(b) `d[1] + c`

(c) `c[-1]`

(d) `c[len(c)]`

(e) `len(d)`

(f) `c[1][1]`

(g) `c[2:3]`

(h) `c[b]`

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

```
def func1(lst, lst2):
    i = 0
    while i < len(lst) and lst[i] < lst2[0]:
        lst2.append(lst[i])
        i += 1

def func2(lst):
    lst2 = []
    for i in range(len(lst)):
        if i % 2 == 0:
            lst2.append(lst[i])

    return lst2

lst = [4, 0, 6]
lst2 = [1, 3, 5]
func1(func2(lst2), lst)
print(lst)
```

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

```
def func3(lst):
    new_lst = []
    for i in range(len(lst)):
        new_lst.append(lst[i][:])

    for i in range(len(new_lst)):
        for j in range(len(new_lst[i])):
            new_lst[i][-1] += new_lst[i][j]
    return new_lst

alst = [[2,4],[5,7]]
print(func3(alst))
```

- (c) (4 points) What is the output of the following program if a file named 'agenda.txt' is the only other file in the same directory? The contents of 'agenda.txt' are shown below.

```
def find_event(base_filename, search_str):  
  
    extentions = ['.csv', '.txt']  
  
    for ext in extentions:  
        try:  
            file = open(base_filename + ext, 'r')  
            lines = file.readlines()  
            for line in lines:  
                if search_str in line:  
                    print(line[:line.index(search_str)])  
            file.close()  
        except FileNotFoundError:  
            print("File not found.")  
  
find_event('agenda', 'April')
```

The contents of the file 'agenda.txt':

```
Department Meeting 2:30pm, 22 March 2018  
Exam 7:00pm, 21 April 2018  
Homework Due 11:00pm, 3 May 2018  
Concert 8:00pm, 24 April 2018
```

3. (8 points) Write a function called `before_zero` that takes as a parameter a list of integers. It should return a list of all integers that appear in the original list before the first occurrence of 0. The function should not modify the original list.

For example:

- `before_zero([1, 2, 0, 4, 5])` returns `[1, 2]` because 1 and 2 occur before the zero.
- `before_zero([6, 7, 6, 0, 8, 0, 9])` returns `[6, 7, 6]` because 6, 7, and 6 occur before the first zero.
- `before_zero([3, 1, 2])` returns `[3, 1, 2]` because there is no zero, so all integers in the list are included.
- `before_zero([0, 1, 2])` returns `[]` because there are no integers before the zero.

4. (10 points) Write a function `parse_city_state_zip` that takes as a parameter `address`, a string containing the city, state and zip code line of an address, for example: `'Hamilton, NY 13346'`.

Your function should parse the address into the various parts and return three values, a string of the city, a string of the state and an int zip code.

For example: `parse_city_state_zip('Hamilton, NY 13346')` would return:

`'Hamilton', 'NY', 13346`

Another example: `parse_city_state_zip('New Woodstock, NY 13122')` would return:

`'New Woodstock', 'NY', 13122`

You **may not** use string methods for this function.

5. This is a two-part question, the second part is on the next page. Part (a) is a helper function for part (b). Part (b) can be completed even if you have not finished part (a) correctly.

(a) (5 points) Write a function called `read_data` that takes a filename (e.g., `nums.dat`) as a parameter. The function will return a list of sublists, where each sublist contains the integers from a single line of the file. The integers on each line of the file are separated by semicolons (`;`). The function should return `None` if the file does not exist.

For example, if the file `nums.dat` contains the following:

```
11;33;55;77
```

```
2;4;8
```

```
5;4;3;2;1
```

`read_data(nums.dat)` will return:

```
[[11, 33, 55, 77], [2, 4, 8], [5, 4, 3, 2, 1]]
```

- (b) (7 points) A climate scientist has hired you to assist them in writing a program that analyzes tree ring widths. The scientist has stored tree ring width measurements in a file. Each line of the file contains the ring width measurements for one tree. The first number on a line is the width of the oldest ring and the last number on the line is the width of the newest ring. The measurements are separated by semicolons (;).

Write a function called `average_width` that takes a filename and a number of years ( $n$ ) as parameters and returns the average ring width among all trees  $n$  years ago. In other words, when  $n$  is zero the function returns the average width of the newest ring on all trees, when  $n$  is 1 the function returns the average width of the second newest ring on all trees, etc. If a tree has fewer than  $n$  rings, it is excluded from the average. If the file does not exist, the function should return -1.

For example, if the file `trees.dat` contains the following:

```
60;65;70;75;80
30;40;50;60;70;80
20;50;80
```

Then, `average_width(trees.dat, 0)` will return `80.0` and `average_width(trees.dat, 3)` will return `57.5`

You are required to use the `read_data` function from part (a) and 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.)