

CSC 111: Intro to Computer Science

Midterm: Spring 2017

Instructor: Sara Mathieson

Completed by:

Friday, March 10 at 5pm

- This exam is to be taken in the Neilson Library during any of their open hours.
- The time limit is **2 hours** unless you received an email saying otherwise. I will be checking all in/out time stamps.
- No communication about the exam with anyone in the class (or outside the class).
- No electronic devices are to be used during the exam, but you may use a 2-sided cheat sheet (8.5" × 11"). Your cheat sheet should be handwritten and created by you (no printed material).
- Discussing the exam, going over the time limit, and using electronic devices are all honor code violations.
- Make sure all your work is contained on these pages (writing on the backs is okay).
- If you are unable to make progress on any part of the exam, tell me what you tried; describe your thought process.
- After completing your exam, reseal the exam using a piece of tape (provided by the library at the circulation desk).

Name	Solutions (sketches)
------	----------------------

Part 1	/25
Part 2	/15
Part 3	/20
Part 4	/15
Part 5	/25
Total	/100

Part 1: Short Answer

3 (a) What numbers are printed when the following code is executed?

```
for i in range(6,-8,-2):
    print(i)
```

6, 4, 2, 0, -2, -4, -6

3 (b) What numbers are printed when the following code is executed?

```
for i in range(-3,-1):
    print(i)
```

-3, -2

3 (c) Describe the issue with the following code, which is attempting to return the sum of the numbers in a list.

```
def my_sum(lst):
    s = 0
    for i in lst:
        s = s + lst[i]
    return s
```

This should be i, since the for-loop is iterating over the elements of the list, not the indices. A variable name besides i would help avoid this mistake.

1 (d) If I invoke the main method below, what will be the value of x?

```
def compute(a,b,c):
    return a*(b-c)

def main():
    a = 1
    b = 3
    c = 2
    x = compute(b,c,a)
```

compute(3, 2, 1)

↳ 3 * (2 - 1)

↳ x = 3

3 (e) What is wrong with this attempt to read a file? How could you modify the variable names to help prevent this type of error?

```
words = open("my_file.txt","r")
for text in words:
    x = words.count("a")
    text
```

I would change:

↳ { words → file
 text → line

- The main issue is that words is the file object, so we cannot call words.count("a"). Should be text.count("a").
- We should also close the file!

- 6 (f) The following lines of code are executed sequentially in the shell. In the right column, write out what is printed *after* each line (some lines of code may not have any output). If a line produces an error, just write "error". Then circle all the lines that make use of *casting*.

line of code	output
>>> x = "3.14"	/
>>> float(x)	3.14
>>> x = x + 1	error
>>> y = float(x)	/
>>> y	3.14
>>> z = int(y)	/
>>> z	3
>>> type(z)	<class 'int'>
>>> type(y)	<class 'float'>
>>> type(x)	<class 'str'>

} as long as you set the right type → okay.

- 6 (g) Write a function in the box below that will take one parameter, a list of full names, and *modify* the list so that it only contains the first names. A few examples are shown below:

```
>>> lst1 = ["Katherine Johnson", "Dorothy Vaughan", "Mary Jackson"]
>>> first_name(lst1)
>>> lst1
['Katherine', 'Dorothy', 'Mary']
>>> lst2 = ["Logan Swanson", "Artemis Metaxa", "Val McCulloch", "Zoe Kendall"]
>>> first_name(lst2)
>>> lst2
['Logan', 'Artemis', 'Val', 'Zoe']
```

```
def first_name(lst):
    for i in range(len(lst)):
        lst[i] = lst[i].split()[0]
```

Part 2: Strings and Functions

The goal for this question is to write a function `capital_index(string)`, which returns the *index* of the first capital letter in a string. If there are no capital letters in the string, the function should print "No capital letters found". Here are a few examples of this function in the shell:

```
>>> capital_index("Spring")
0
>>> capital_index("brEaK")
2
>>> capital_index("hello")
No capital letters found
>>> capital_index("computeR")
7
>>> capital_index("sciEnCE")
3
```

Below is a start at creating this function. First a helper function is defined to see if a single letter is capital or lowercase. Answer the following question about this function and complete the code.

```
# returns whether or not a single letter is a capital
# example: "a" will return False and "A" will return True
def is_capital(letter):
    if letter == letter.upper():
        return True
    return False

def capital_index(string):
    for i in range(len(string)):
        if is_capital(string[i]):
            return i
    print("No capital letters found")
```

- ① (a) What is the *type* of the return value of `is_capital(letter)`?
- ⑭ (b) Fill in the code for `capital_index(string)` above. Your code must make use of the helper function `is_capital(letter)`, and its behavior should match the shell output shown above.

Part 3: Mystery Lists

Analyze the code below and answer the following questions.

```
import random

def mystery_method1(n):
    function
    r_lst = []
    for i in range(n):
        r_lst.append(random.randint(0,19))
    return r_lst

def mystery_method2(lst):
    function
    for j in range(len(lst)-1):
        if lst[j] > lst[j+1]:
            return False
    return True

def main():
    num = eval(input("Enter a number: "))
    my_lst = mystery_method1(num)
    result = mystery_method2(my_lst)

    print(my_lst)
    print(result)

main()
```

② (a) List all the *formal* parameters of the helper functions above.

n, lst

② (b) List all the *actual* parameters of the helper functions above.

num, my_lst

*considering only
mystery methods*

② (c) Is randint a variable or a function?

a function

- ④ (d) What does `mystery_method1` do? Describe this function in words, including the *type* of the parameter (input) and return value (output).

`mystery_method1` creates and returns a list of random numbers. The input is an integer, which specifies how many random numbers to generate. The output is the list of random numbers, all between 0 and 19 inclusive.

- ④ (e) What does `mystery_method2` do? Describe this function in words, including the *type* of the parameter (input) and return value (output).

`mystery_method2` checks whether or not the list is sorted (i.e. in order). It must be sorted ascending (low to high) for a True return value. The input is a list of numbers and the output is a boolean (True for sorted & False otherwise).

- ③ (f) Say after `mystery_method1` is called, `my_lst` has the value `[4, 3, 7]`. What is result? Briefly explain your answer.

False not in ascending order.

After we find that $4 > 3 \rightarrow$ return False and stop.

- ③ (g) Same as part (f), but what if `my_lst` as the value `[0, 15, 17]`? Briefly explain your answer.

True list is sorted in ascending order.

Part 4: Loops and Printing

The following examples show an inverted triangle being printed after the user enters an integer. Write and call a main method to produce this result. Notice that if the number is 3, there are 7 stars in the first row, and if the number is 5, there are 11 stars in the first row. Your main method should be general for any integer.

```
>>> ===== RESTART =====
>>>
Enter a number: 3
*****
 *  *
 * *
 *
>>> ===== RESTART =====
>>>
Enter a number: 5
*****
 *      *
 *      *
 *      *
 *      *
 *      *
 *
>>>
```

```
def main():
    n = eval(input("Enter a number: "))
    print("*" * (2n+1))
    for i in range(1,n):
        print(" " * i + "*" + " " * (2*(n-i)-1) + "*")
    print(" " * n + "*")

main()
```

Part 5: Changing Variables

In class we have seen how to swap the values of two variables, and in homework we have seen how to swap the values of two elements in a list. In this question, the goal is to swap the values of *three* variables, so that each variable ends up with the value of the “previous” variable. For example, if $x = 6$, $y = 3$, and $z = 1$, then the end result should be $x = 1$, $y = 6$, and $z = 3$.

- 5 (a) The following code shows a first attempt at this process. Fill in the table below, showing what will happen *after* each line is executed. The first row has been filled in with the initial values from the example above.

code	x	y	z	temp1	temp2
	6	3	1	-	-
temp1 = x	6	3	1	6	-
temp2 = y	6	3	1	6	3
x = z	1	3	1	6	3
y = x	1	1	1	6	3
z = y	1	1	1	6	3

- 5 (b) Explain the issue with the code above, and rewrite the code with a few modifications so that it successfully swaps these three variables.

Temporary variables are being assigned, but not used to update the x, y, z variables correctly.

code	x	y	z	temp1	temp2
temp1 = x	6	3	1	6	-
temp2 = y	6	3	1	6	3
x = z	1	3	1	6	3
y = temp1	1	6	1	6	3
z = temp2	1	6	3	6	3

modified from above



- 5 (c) Can you perform this three-variable swap with only *one* temporary variable? If no, explain the issue with such an approach. If yes, provide the code and a table like the one above.

	x	y	z	temp
temp = x	6	3	1	6
x = z	1	3	1	6
z = y	1	3	3	6
y = temp	1	6	3	6

Yes!

matches our goal

★

- 5 (d) Would it be possible to write a *function* to swap three variables? i.e. would it be possible to write a function that produces the output below? Explain your answer and reasoning, using the concept of mutable vs. immutable types.

```
>>> x = 6
>>> y = 3
>>> z = 1
>>> three_way_swap(x,y,z)
>>> x
1
>>> y
6
>>> z
3
```

No integers are immutable, so we cannot change their variables inside a function. We could return new values and reassign the variables, but we cannot obtain the output to the left.

- 5 (e) Write a function that will return a new string with the *i*th character and *j*th character swapped. Example: `string_swap("spring", 1, 5)` should return "sgrinp". You may assume that *i* is less than *j*. Why must this function return a *new* string?

```
def string_swap(string, i, j):
    return string[:i] + string[j] + string[i+1:j]
        + string[i] + string[j+1:]
```

strings are immutable! Cannot modify a string.

More space (tell me which problem this is for), or draw me a picture.

For fun: in Part 1 (g), who are the two lists of names referring to?

- ① The African American female NASA scientists/engineers celebrated in the movie "Hidden Figures."
- ② The TAs from our four lab sections.