# CMSC 201 - Fall 2023 Midterm I

## Circle your section number and TA.

| | | | |
|---|---|---|---|
| 11 - Anna Pham | 12 - Alexander G. | 13 - David P. | 14 - Rusham B. |
| 15 - Gabe A. | 16 - Malcolm D. | | |
| 21 - Alexander J. D. | 22 - Matthew P. | 23 - Kelvin Z. | 24 - Nathan D. |
| 25 - Khushika Shah | 26 - Claire Kim | 27 - Ky DeSilva | |
| 31 - Adrian Maldonado | 32 - Lyna Beraich | 33 - Claire Kim | 34 - Rodolfo A-R |
| 35 - Raine Gibson | 36 - Hannah Oskam | 42 - Hikaru B. | 43 - Gia O.S. |
| 44 - Hieu T. | 45 - Sean H. | 46 - Pranav S. | |
| 51 - Wojciech L. | 52 - Wojciech L. | 53 - Papa M. | 54 - Drake T. |
| 55 - Hannah Oskam | 56 - Leann A. | 58 - Niraj D. | |

## Write your name (neatly!) on the line below.

Shane

First Name    Last Name

sdonahue

Student ID, which you use for GL

## Undergraduate Honors Statement

In this course, each student assumes the responsibilities of an active participant in UMBC's scholarly community, in which everyone's academic work and behavior are held to the highest standards of honesty. Cheating, fabrication, plagiarism, and helping others to commit these acts are all forms of academic dishonesty, and they are wrong.

## Multiple Choice: 24 points (2 points each)

Choose the correct answer by clearly filling exactly one circle per problem.

1. Which statement evaluates to **True**?

   ○ **True** and (**False** or **False**) → True and (False) → False

   ● **False** or (**True** and not **False**) → False or (True and True) → True

   ○ **True** and **False** → False

   ○ not(**False** or not **False**) → not (False or True) → not (True) → False

2. Which loop cannot be used to modify the element of a list?

   ● for-each (for x in my_list) → uses elements (pass by value)

   ○ for-i (for i in range(len(my_list)))

   ○ while (while i < len(my_list))

   } uses. indexes (pass by reference)

3. Which variable is legal in Python (will **not** cause a syntax error)?

   ○ 32number

   ● hello_there

   ○ $perl_is_good

   ○ !goodVariable

   no numbers or symbols at start of var >:C

4. When running a while loop, in order to avoid an infinite loop you must:

   ○ Set all variables to zero. → no...

   ○ Make sure none of the variables inside the loop changes. → would not help

   ● Change a variable or take user input to see if the while condition changes.

   ○ Not do any input inside of the loop. → largely irrelevant

5. To check if an integer $x$ is divisible by 7 you would use:

   ○ $x$ % 2 == 0

   ○ 7 % $x$ == 0

   ● $x$ % 7 == 0

   ○ 7 // 2 == 1

   % is modulus → gives remainder

   If remainder is zero, it's cleanly divisible!

6. Why is / different from //?

   ○ Because x // y gives back a remainder rather than the quotient.

   ○ Because // is the symbol for a comment.

   ○ Because // is the symbol for modulus.

   ● Because // is integer division rather than floating point division.

   // is integer division in Python.

   (see lecture notes for full list of operations)

7. If you know how many times you want to iterate, which of the following will you (most likely) use?

● For loop. → for loops → iterate through list or range, with predetermined length

○ While loop.

○ If statement. → does not iterate

○ Print loop. → not a real thing

while loops: run a usually unknown number of times (based on boolean condition)

8. How many times will this loop execute for $x$ in range$(3, 12, 2)$?

start (inclusive) stop (exclusive) step

○ 9

○ 3

○ 7

● 5

3, 5, 7, 9, 11.

9. When “Zero” is cast to an integer, what happens?

"Zero" is a string.

○ The integer that results is 0 as an integer.

○ The program will hang (not terminate but not crash).

● An error will occur which terminates the program.

○ The integer that results is 0.0 as a float.

int("1") → 1 ✓ OK

int("one") → error

int("zero") → error

10. Which of the following is **True**?

● ’he’ in ’hello world’. ✓ true (in is substring check)

○ 3 in [1, 2, 5] → no, 3 not in list

○ ’sam’ in ’swarm’ → no, 'sam' not in 'swarm'

○ 0 in ’Zero’ → no, int (0) not in string "Zero"

11. **At most** how many elif statements **will execute** in a single if statement?

○ 0

● 1

○ ∞

○ −3

if
elif
elif
...
else

→ only one will run. mutually exclusive!

12. How many elif statements **can you write** in a single if statement?

○ 0

○ 1

○ 17

● As many as you can type in.

But you can write as many as you'd like =)

## List Manipulation: 6 points (2 points each)

For the following three problems consider this list:

robots = [”dalek”, ”data”, ”R2D2”, ”K2SO”, ”Marvin”, ”HAL 9000”]

0 1 2 3 4 5

For each of the problems here, write what is output in the space provided. If nothing is output, write “NO OUTPUT” and if there is an error of any kind from SyntaxError, IndexError, ValueError, just write “Error.”

**Each part should be considered independently, meaning you should restart from the starting list every time.**

6 - 4 = 2

13.
```
print(robots[5], robots[len(robots) - 4])
```

last element

HAL 9000 R2D2

14.
```
robots.remove('R4')
print(robots)
```

→ not in list

Error

15.
```
robots.append('Bender')
print(len(robots))
```

→ add new element, list is now one longer (6 → 7)

7

## Creating Boolean Expressions: 4 points (2 points each)

16. Write an expression that is True if and only if a string `food` contains ‘cheese’ (as a substring) and `drink` is not ‘water’.

"cheese" in food and drink != "water"

17. Write an expression that is True if and only if `x` is at least 30 and `y` is between 1 and 10 (inclusively).

x >= 30 and (y >= 1 and y <= 10)

## Short Answers: 6 points (2 points each)

18. What are the two possibilities that can happen based on different inputs of x?

```
x = int(input('Input int: '))
while x > 0:
  print('hello')
```

if x > 0 → run forever
(loop condition always true)

otherwise never print anything

19. Explain what happens when we replace an if with an elif statement instead. What is the difference between the two? For instance consider the code below:

```
num = int(input('>>'))
if num > 10:
  print('num is larger than 10')
if num > 20:
  print('num is larger than 20')
```

elif will make the if/elif mutually exclusive, so it will only print one or the other. Right now with two if statements, they could both run (e.g., num = 30).

20. Fill each blank with the a word conveying the meaning of the argument in the range.

| | | | |
|---|---|---|---|
| range( | start | stop | step ) |
| range( | start | stop ) | |
| range( | stop ) | | |

## Order of Operations: 3 points (1 points each)

Evaluate the following expressions.

21. Let $x = 4, y = 5$, then:

```
3 * (x + y) < y ** 2 - x
```

$$\underbrace{3 * \underbrace{(x + y)}_{9}}_{27} < \underbrace{\underbrace{y ** 2}_{25} - x}_{21}$$

27 < 21

$\boxed{\text{False}}$

22. `True and (False or not False)` → True and True

$$\underbrace{\texttt{not False}}_{\text{True}}$$

$\boxed{\text{True}}$

23. `not [] and not (5 or 'a')`

Falsey

Truthy

not False and not True

True and False

$\boxed{\text{False}}$

## Code Evaluations: 6 points (2 points each)

Evaluate each code snippet below and answer the question asked. You may assume that the code is syntactically correct. Make sure that you pay attention to the formatting of the output when you write down your answer. **Mark your final answer clearly by boxing it.**

24. When the code is run, what is the output?

```
halving = 50
while halving > 0:
  halving //= 2
  print(halving)
```

← integer division

divide by two while halving > 0, print each time

```
25
12
6
3
1
0
```

25. When the code is run, what is the output?

```
s = 'hello'
n = 15
if n == 20 and s == 'hello':
  print("Both are true.")
elif n == 15:
  print("15 is equal to 3 times 5")
elif s == 'hello'
  print("Well Hello There")
else:
  print("Obi Wan Kenobi, You're my only hope")
```

→ False

→ True

15 is equal to 3 times 5

after first condition in if/elif/else is executed the rest are skipped

26. When the code is run, what is the output?

```
for i in range(16, 2, -4):
  print(i)
```

start stop step

```
16
12
8
4
```

## Debugging: 12 points (1 point per line, 2 points per correction)

The code below has 6 errors. The errors may be syntax errors or logic errors, consider all errors that occur in one line as a single error; examine the code carefully to find them. Indicate each of the errors you find by writing the line number and correction in the space provided below. Each error can be solved with a single fix, and should not be counted twice. Comments do not contain errors. The direction of the string symbols are not an error.

27. You must find and correct 4 of the errors. (You cannot fix more than 4 errors!)

```
 1  """
 2    We will remove all of the bad symbols, which are in the
 3    bad_symbols list from the strings in the_list.  We will
 4    count the number of removals and report them.
 5  """
 6  # the lists do not contain errors
 7  bad_symbols = ["!", "@", '#', '$', '%', '^', '&', '*',
 8            '(', ')', ',', '.']
 9  the_list = ['hello', '&(*Byte', 'Kilobyte!', 'S$u@s!h#i']
10  # start here
11  new_list = []
12  removed_symbol = 0                                        missing colon
13  for the_string in the_list
14    new_string = ''                                                       be "not in"
15    for x in range(len(the_string)):                    should
16      if the_string[x] in bad_symbols:
17        new_string += the_string[x]
18      elif: should be else
19        removed_symbol + 1      needs to be += to save
20    new_list.append(the_string)      new value
21                                     should be new_string
22  print(', '.join new_list )
23  print(f'We removed {removed_symbol} bad symbols')
```

| Line Num. | Your correction for that error. |
|---|---|
| 15 | negate condition (not in) |
| 17 | use else, not elif |
| 19 | use += to save new count |
| 20 | append new_string, not the_string |

## Programming Problem No. 1: 7 points

28. Input a list of integers (you are allowed to assume that the user will only type integers) until the user types zero. Output a list where the even numbers are squared from what was input but the odd numbers are left alone.

    For instance if you input the numbers 21, 13, 8, 3, 2, 6, 17, 0 then the program will output [21, 13, 64, 3, 4, 36, 17].

```
if __name__ == ’__main__’:
    num_list = []
    user_int = int(input())
    while user_int != 0:
        if user_int % 2 == 0:
            num_list.append(user_int ** 2)
        else:
            num_list.append(user_int)
        user_int = int(input())
    print(num_list)
```

if even, square it

else, just add it to list

## Programming Problem No. 2: 7 points

29. Write a program after the beginning code that takes the string_list and determines the string with the maximum number of ”a” characters. For instance if the user enters ”archibald” ”aardvark” ”sandal” ”stop” then aardvark will be printed since it has the most ”a” characters. Notice we have already converted the strings to lower case so you don’t have to worry about that.

```
if __name__ == '__main__':
  string_list = []
  s = input("Enter a string: ")
  while s != "stop":
    string_list.append(s.lower())
    s = input("Enter a string: ")
  # begin your code here
  max_a_str = ""
  max_as = -1

 for string in string_list:
    a_count = 0
    for letter in string:
        if letter == "a":
            a_count += 1

    if a_count > max_as:
        max_as = a_count
        max_as_str = string

print(max_as_str)
```

← empty string

← -1 so that any string will be our new max

count a's in current string

if we had the most a's, save our string to max_as_str