CMSC 201 Spring 2018 Name_____

Midterm Review Worksheet

This worksheet is ***NOT*** guaranteed to cover every topic you might see on the exam. It is provided to you as a courtesy, as additional practice problems to help you study. You should also be reviewing the course notes and assignments as part of your preparation for the exam.
No answers will be provided, either via an answer sheet or by the TAs, for the questions on this worksheet. You are encouraged to work with other students in the class to confirm your answers and solidify your understanding of the material.
The questions on this worksheet may be more difficult/tricky than ones you would see on an exam.

1. Name and describe the four control structures used to control the flow of a Python program.
2. What is the **`def`** keyword used for in Python?
3. Is it possible to write a program that prints 3 lines of output without having 3 separate **`print()`** statements? If so, what would that look like?

4. Write a simple Python program that (a) assigns a list of the prime numbers less than 10 to a variable called **`myList`**, (b) prints the length of this list, (c) prints each element of **`myList`** by using a **`while`** loop, and (d) prints each element of **`myList`** in reverse using a **`while`** loop.
   (You can hard-code step (a) by creating **`myList`** *and initializing it with the numbers 2, 3, 5, and 7.)*

5. Write Boolean expressions for each of the questions below.
   You do not need to include the "`if`" keyword in your answer.
   For example, if the question is "An expression that evaluates to True only if the integer number is odd and not 19." the answer would be **`number % 2 == 1 and number != 19`**
   a. An expression that evaluates to **`True`** only if the integer **`num`** is positive (greater than zero) and the string **`animal`** is equal to "dog".
   b. An expression that evaluates to **`True`** only if the Boolean **`bool1`** is False, the Boolean **`bool2`** is True, and the Boolean **`bool3`** is not True.
   c. An expression that evaluates to **`True`** only if the integer **`a`** is greater than the integer **`b`**, the integer **`b`** is less than 7, the integer **`a`** is not equal to 5, and the integer **`c`** is positive.
   d. An expression that evaluates to **`True`** only if the integer **`number`** is not odd and the integer **`number`** is between 11 and 99, inclusive.
   e. An expression that evaluates to **`True`** only if the string **`name`** is lowercase.
   f. An expression that evaluates to **`True`** only if the string **`name`** is less than 12 characters, and starts with a "C" or a "2".

6. Give three examples of legal **variable names** and three different examples of illegal variable names in Python. For the illegal variable names explain why they are illegal.

7. Write a snippet of Python code that continuously takes input from the user using a **`while`** loop, and adds that input to the end of a **list**. When they enter "**`quit`**" the program should print the list and terminate.
8. Write a snippet of Python code that reads an **integer** from the user, and then computes that integer's **absolute value**. The absolute value of a number x is defined as

$$|x| = \begin{cases} x & \text{if } x \geq 0 \\ -x & \text{otherwise} \end{cases}$$

9. Answer the questions below, given the following strings. (We've made the strings big enough so that you can write the indexes of the letters to help you count. It will be like this on the exam as well.)

```
daya      = "I don't wanna sit still look pretty"
glimpse   = "life wasn't easy, but living it was"
oliver    = "every boulevard is a miracle mile"
callMeAl  = "I can be your long-lost pal"
perry     = "baby you're a firework"
```

What do the following pieces of code evaluate to?	How would you get each of the following substrings from the indicated string?

`print(perry[ len(perry)-8 : ])`
`print(callMeAl[6 : 6 + 7])`
 	
`print(daya[24:28] + perry[4:12] +`
`  daya[ len(daya) - 6 : ])`
 	
`text = perry[:4]`
`print(text + text + text)`
 	
`print(glimpse[6], oliver[6],`
`  callMeAl[2])`
 	
`var  = (len(perry) - 1) // 10`
`gLen = len(glimpse)`
`print(perry[5 : 8],`
`  callMeAl[var : var + 3],`
`  daya[2 : 4],`
`  glimpse[gLen - 6 : gLen - 4])`
	`"wanna" from daya`
	
	`"lost" from callMe`
	
	`"fire" from perry`
	
	`both "was" from glimpse`
	
	`"every" from oliver`
	
	`the apostrophes in daya, glimpse, and perry`
	
	`the hyphen in callMeAl`
	
	`"miracle mile" in all caps from oliver`

10. Write a program that outputs <u>every other</u> element in a list called **`studentNames`**.

11. Write a program that allows the user to input 5 numbers and then uses a function called **`addAll()`** to add them all together and output the sum. (It is your decision if the numbers are passed in as a single list, or as five separate variables.)

12. Write a **`while`** loop to perform each of the following actions.
    a. Iterate over a list
    b. Validate input from a user
    c. Print out the numbers for a countdown
    d. Loop until a specific condition is met

13. Complete the code by **filling in the blanks** for each question below. (The length of the blank doesn't matter.)

a. Print out the contents of the list `cars`.

```
cars = ["honda","ford","porsche","tesla"]
index = 0
while _____  _____  _____:
    print(_____)
    _____
```

b. Read in 10 integers from the user.

```
numberList = []
while _____  ____ 10:
    userNum = _____(_____("Enter a num: "))
    numberList._____(_____)
```

c. Add the numbers 5 through 10 together.

```
total = _____
count = _____
while _____  ____  _____:
    total = _____ + _____
    count = _____
print("The total of 5 through 10 is", _____)
```

14. For each of the pieces of code below, circle and **explain any errors** you find. (There may be more than one in a single statement!) You can assume that variables are initialized and contain what their names indicate (*e.g.*, `intCounter` is an integer, etc.).

a.
```
main():
    course == "CMSC 201'
```

b.
```
def diff(num1, num2)
    ans = num1 - num2
    return ans
def main():
    diff(51, 23)
    print(ans)
```

c.
```
candyPrice = 1.25
candyPrice = 1.25
myMoney = int("twenty")
moneyLeft = myMoney - candyPrice
print("I have $" + moneyLeft + " money after buying candy.")
```

d.
```
def printStatement(num):
    print("num is" + num)
def main():
    printStatement(5) * 3
```

e. `name = input(print("Please enter your name:"))`

f. The following code attempts to compute the equation `m * x + b` and assign that value to `result`.

```
result = mx + b
```

g. This code increases `x` every iteration until it is greater than `y`.

```
x = 5
y = 5
while x <= y:
    x = x + 1
    y = y + 1
print ("Done!")
```

15. What is the **output** of each small Python program below?

(a)
```
x = 3
y = 4
print (x)
x = 4
```

(b)
```
x = 3
while x < 6:
    x = x + 1
print (x)
```

(c)
```
x = 6
y = 5
if x >= y:
    x = x - 2
print (x)
```

(d)
```
tc1 = 100
tf  = (9/5) * tc1 + 32
tc2 = (tf - 32) * 5/9
print (tf)
print(tc2)
```

(e)
```
x = 0
while x < 5:
    if x % 2 == 0:
        print("even: ", x)
    else:
        print("odd:  ", x)
    x = x + 1
```

(f)
```
x = 1
i = 1
while x <= 4:
    x = x * i
    i = i + 1
    print (x)
```

16. Answer the following questions about **data types in Python**.
    a. Which built-in function do you use to cast an integer into a floating-point number?
    b. Which built-in function do you use to cast a string into an integer?
    c. What happens when you try to convert a string into an integer but the string is not a number?
    d. When would you want to convert an integer into a string? In other words, in what cases would a programmer need to do such a conversion? Demonstrate with an example.
    e. What is the most appropriate data type to represent decimal numbers in Python?

17. What is the output of the code below?

```
wishList = ["PlayStation 4", "Puppy", "Chocolate"]
wishList.append("Puppy")
wishList[0] = "XBox One"
wishList[0:3]
print(wishList)
```

18. What is the value of **mystery** after this sequence of statements?

```
mystery = 1
mystery = 1 - 2 * mystery
mystery = mystery + 1
```

19. What is the value of **mystery** after this sequence of statements?

```
mystery  = 3
mystery  = mystery + 1
mystery += 3 * 5 - 1
```

20. What is the value of **mystery** after this sequence of statements?

```
mystery  = 5
mystery  = (15 - mystery) % 2
mystery += 1
```

21. Circle and correct at least five errors and identify what kind of error it is (logical or syntax):

```
def outPrint(strName, age, gpa):
    print("The student's name is:", name)
    print("Their age is: " + int(age))
    print("Their gpa is: " + float(age))

def main():
    name       = input("Enter the student name: ")
    int(age)   = input("Enter the student's age: ")
    float(gpa) = input("Enter the student's gpa: ")
    outPrint(name, gpa, age)
main()
```

22. Find and correct at least eight errors in the code below. They may be syntax or logic errors. (FYI: This code is very broken, and requires a lot of "simple" fixes to be able to run.)

The rules of blackjack are that you want to get as close to a total of 21, without going over. Cards are dealt randomly, so although there is some strategy, it is largely a game of chance.

```
def main():

     BLACKJACK = 21
     1st_hand = ["13", "9"]

     cardSum = 0
     index   = BLACKJACK
     while index < cardSum:
          cardSum = cardSum + 1st_hand[index]

     if cardSum >= BLACKJACK:
          print("You lost!")
     else if cardSum < BLACKJACK:
          print("You can hit, or check.")
     else:
          print("You beat the dealer!")

     1st_hand.removeAll()
```

23. Find and correct at least eight errors in the code below. They may be syntax or logic errors.

```
main():
    # hours worked per weekday (Mon-Fri)
    hours = [6.8, 5.7, 4.9, 8.0, 3.2]

    # calculate total
    total = 1
    index = 0
    while index < len(total)
        total = total + hours[h]
        index = 1

    # calculate average
    avg = total % range(hours)

    # print total and average
    print("This worker worked", total, "hours this week.)
    print("For an average of", total, "hours per day.")

main()
```

24. Find and correct at least six errors in the code below. They may be syntax or logic errors.

```
# digits and their English equivalent
DIGITS = <"zero", "one", "two",   "three", "four",
          "five", "six", "seven", "eight", "nine">

def main():
    num = int(input("Please enter a positive integer: "))

    # input validation for only accepting positive numbers
    while (num > 0):
        print("The number", num, "is not a valid choice.")
        num = int(input("Enter a number 1 or higher: "))

    # create variables to use below
    num = copy
    string = ""

    # go through and convert each digit to English
    while copy > 0:
        string = string + " " + DIGITS(copy % 10)
        copy = copy / 10

    print("The number", num, "is", string)
```

25. Define each of the following terms.
(This is meant to help test your **<u>understanding</u>** of the terms, not whether you can recall the "correct" definition from the slides or book.)

1. Algorithm
2. Argument
3. Assignment Operator
4. Bang Equals
5. Boolean
6. Boolean Flag
7. Bracket (*e.g.*, square brackets)
8. Branching
9. Bug
10. Call (e.g., function)
11. Case Sensitive
12. Casting
13. Code
14. Comment
15. Comparison Operators
16. Concatenation
17. Conditional
18. Constant
19. Debugging
20. Decisions
21. Definition (e.g., function)
22. Delimiter
23. "Equivalent to"
24. Float
25. Formal parameter
26. Function
27. Index
28. Infinite Loop
29. Initialize
30. Input Validation
31. Integer
32. Integer Division
33. Interactive Loop
34. Interpreter
35. Iterate
36. Keyword
37. List
38. Literal
39. Logic Error
40. Logical/Boolean Operators
41. Looping
42. Magic Numbers
43. Main
44. Membership Operator
45. Method
46. Modulus (or Modulo/Mod)
47. Nested (*e.g.*, loops)
48. Operator (*e.g.*, assignment)
49. Program
50. Pseudocode
51. Return
52. Scope
53. Sentinel Loop
54. Sequential
55. Slicing
56. String
57. Syntax
58. Syntax Error
59. Value
60. Variable
61. Whitespace