CMSC 201 Fall 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. Answers will be provided a day or two before the first exam.

DO NOT WAIT UNTIL ANSWERS ARE RELEASED TO START THIS WORKSHEET.

You are encouraged to work with other students in the class to confirm your answers and solidify your understanding of the material. You are also encouraged to seek help from TAs during office hours should you be stuck on question.
The questions on this worksheet may be more difficult/tricky than ones you would see on an exam.

1. Name and describe the three control structures used to control the flow of a Python program.
2. Describe two situations where nested decision structures would be useful in a program.
3. Is it possible to write a program that prints 25 lines of output without having 25 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. Illegal ***does not*** mean against coding standards.
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. Write a program that outputs every other element in a list called `studentNames`.

10. Write a program that allows the user to input an integer and outputs all positive factors of that integer (a factor cleanly divides the number).

11. 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

12. 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", _______)
```

13. 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.
```
while counter = 5:
      print("The counter is at", counter)
      counter += 1
```

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

d.
```
x = int(input(“Enter a number:”))
if x < 10:
     print(“Really? Dream a little bigger next time.”)
     else:
          print(“Not a bad choice.”)
```

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!")
```

14. 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)
```

15. 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?

16. What is the output of the code below?

```
wishList = ["PlayStation 4", "Puppy", "Chocolate"]
wishList.append("Puppy")
wishList[0] = "Nintendo Switch"
print("Bubblegum" in wishList)
print(wishList)
```

17. What is the value of `mystery` after this sequence of statements?

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

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

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

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

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

20. Circle and correct the errors below and identify what kind of error it is (logical or syntax):

```
userNum = input(“Enter a number”)
result = userNum % 2
if userNum != 0:
     print(“Your number is even”)
else
     print(“Your number is odd)
```

21. Find and correct at least eight errors in the code below. They may be syntax or logic errors. (FYI: This code is <u>very broken</u>, 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 <u>over</u>. 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()
```

22. 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()
```

23. 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
    1copy = num
    print("The number", num, "is ", end="space")

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

    print()
```

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

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