CMSC 201 Spring 2017 Name_____

Final Review Worksheet

This worksheet is ***NOT*** guaranteed to cover every topic you might see on the final. It is provided to you as a courtesy, for students who want practice problems to help them with their studying. You should review the course notes and assignments as part of your preparation for the final.
No answers will be provided for the questions on this worksheet. You are encouraged to work with other students in the class to confirm your answers and cement your understanding of the material. Some of the questions on this worksheet are more difficult or tricky than ones you would see on an exam – you have much more time, as well as TA assistance available, when working on the review.

1. Suppose you know myInt is an integer and intString is a string representing an integer. For example, myInt is 3 and intString is '24'. Write a function that takes them both in, and prints out the arithmetic sum of the two. In the example given, 27 would be printed.

2. Explain the difference between **`read()`**, **`readline()`**, and **`readlines()`**. Give an example of when you might use each.

3. What would the output from the following code be?

```
counter = 0
for i in range (10):
    print("X" * (counter + 2))
    counter += 1
```

> Don't forget that you can always test code by running it in the Python interpreter, or by saving and running it as a Python file!

4. What would the output from the following code be?

```
def addThree(num):
    return num + 3
def doAThing(thing1, thing2):
    print(thing1 * thing2)
    print( addThree(thing2) )

def main():
    doAThing('x', 4)
    doAThing(addThree(2), 6)
main()
```

5. What would the output from the following code be?

```
lyrics = "What life looks like from up above and down below"
print( lyrics[ : ])
print( lyrics[    : 9  ] )
print( lyrics[ 6  : 8  ] )
print( lyrics[ 26 : 29 ] + lyrics[ 35 : 43 ])
print( lyrics[ (len(lyrics) - 10) : len(lyrics)] )
```

6. Use the **`range()`** function to create the following lists of numbers:
   a. `[5, 20, 35, 50]`
   b. `[-8, -5, -2, 1, 4, 7, 10, 13, 16]`
   c. `[0, 1, 2, 3, 4, 5, 6, 7]`
   d. `[88, 85, 82, 79, 76, 73, 70, 67]`

7. Convert the following **binary numbers to decimal**.
    a. 0011 0011
    b. 1011 1110
    c. 1111 0000

8. Convert the following **decimal numbers to binary**.
    a. 126
    b. 83
    c. 29

9. The code below has seven errors for you to find and correct. If the code worked correctly, it would ask the user for their rating on different movies, save their response, and print out the final information at the end.

```
 1  def rating(ratings, movies)
 2      for m in movies.keys():
 3          newRating = input("Rate ", m, " starring ",
 4                            movies[m], ": ")
 5          ratings.append( (m, movies[m], newRating) )
 6      return newRating
 7
 8  def main():
 9      stars = ["Gladiator": "Russell Crowe",
10               "Titanic": "Leonardo DiCaprio",
11               "Alien": "Sigourney Weaver"]
12      myList = {}
13
14      stars["Zombieland"].append("Jesse Eisenberg")
15      print("For each movie, please rate from 1-10")
16      ratings = rating(myList, stars)
17
18      for j in ratings:
19          print("You rated", j[2], "starring", j[1], "a", j[0])
20
21  main()
```

10. Write a snippet of code that gets a number from the user and prints "High" if it is over 100, "Low" if it is less than 50, and "In between" otherwise.

11. Write a snippet of 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 twice (once in the given order, and once in reverse) before terminating.

12. Write a snippet of code that gets a list of integers from the user, and then uses bubble sort to sort those numbers inside the list. At the end, it should print out how many swaps it made, and how many passes it took before the list was sorted (you should include the final "check" pass).

13. Write a function that uses *recursion* to reverse a string.

14. Write a function that uses *recursion* to test if a number is prime.

15. Write a function that uses *recursion* to find the maximum number in a list.

16. Write a function that uses *recursion* to get valid input from the user. (HINT: You would call the function again instead of using a while loop to re-run when the input is invalid.)

17. Write a function that creates and returns a 2d list, where the contents count up, while the size of the "inner" lists goes down in size. For example, with an input of 4, the list would look like
```
[ [1, 2, 3, 4], [5, 6, 7], [8, 9], [10] ]
```

18. Write a function that takes in an integer and determines if it is a power of 2, returning **`True`** or **`False`**. (Powers of 2 include 1, 2, 4, 8, 16, 32, 64, etc.)

19. The recursive Fibonacci function we created in class runs very slowly, taking over 2 and a half hours to calculate the 50th Fibonacci number. Write a function that makes use of a dictionary to store the calculations that were already performed. The keys should be the number we're requesting (*e.g.*, the 50th number, 49th number, etc.) and the values should be the answer for each (*i.e.*, the value of the 49th Fibonacci number should be stored with the 49th key).

20. Study with friends! Write up and test a piece of code for one of problems above. Then, remove some of the pieces and replace them with blanks. Give it to your friend to fill in, and have them do the same for you. Or, you could add in some errors to the code, and challenge them to fix it.

21. For each of the short programs below, circle and **explain any errors** you find. (There may be more than one in a single statement! A statement may also be error-free.) You can assume that variables are initialized and contain what their names indicate (*e.g.*, **`int1`** is an integer, etc.)

a.
```
def addTwoNumbers( int(num1), int(num2) ):
    return ans
    ans = num1 + num2
def main():
    added = addTwoNumbers(4, 5, +)
    print( added )
main()
```

b.
```
def diff(num1, num2):
    num1 -= num2
    return num1
def main():
    1_int = 5
    int#2 = 7
    diff(1_int, int#2)
main()
```

c.
```
def printStatement(num1):
    print( str(num1) * int(num1) )
def main():
    print( printStatement(5) )
```

22. 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. Argument (or Parameter)
3. Attribute
4. Base Case
5. Binary
6. Boolean
7. Branching
8. Bug
9. Case Sensitive
10. Class
11. Code
12. Comment
13. Concatenation
14. Conditional
15. Constant
16. Constructor
17. Debugging
18. Dictionary
19. Error (e.g., logic error)
20. Incremental Development
21. Index
22. Infinite Loop
23. Inheritance
24. Input and Output
25. Integer
26. Integer Division
27. Interpreter
28. Iterate
29. Keyword
30. List
31. Logic
32. Loop
33. Main
34. Method
35. Modularity
36. Modulus (or Modulo/Mod)
37. Mutable (and Immutable)
38. Nested (e.g., loops)
39. Object
40. Operator (e.g., assignment)
41. Parent / Child
42. Program
43. Pseudocode
44. Recursion
45. Recursive Case
46. Return
47. Scope
48. Selection
49. Sequential
50. Sorting
51. String
52. Syntax
53. Truth Table
54. Tuple
55. Value
56. Variable
57. Whitespace

23. What do each of the following Linux commands do? Give an example of how you would use each.
    1. **`cd`**
    2. **`ls`**
    3. **`pwd`**
    4. **`emacs`**
    5. **`python`**
    6. **`cp`**
    7. **`mv`**
    8. **`mkdir`**
    9. **`submit`**

24. Given an example of each of the following types of errors: syntax, runtime, type, and logic.

25. You should also know the following concepts, topics, and/or how to code them:
    a. File I/O
        i. Including how to use split() and strip() correctly
    b. Selection Sort, Bubble Sort, and Quicksort
        i. (Don't need to code them, but should know how they work and their run times)
    c. Linear search and binary search (again, should know their run times)
    d. Creating and printing 2D and 3D lists
    e. Creating, updating, and removing elements of a dictionary
    f. Recursion!
        i. (If you skipped or didn't understand Labs 11 or 12, you should look at them)
    g. Recursion!

The final covers more topics, and more difficult topics (recursion, 3D lists, file I/O, searching and sorting) than the midterm. It will be a more difficult exam!