Final 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 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. 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):
    for j in range(counter + 2):
        print("X", end ="")
    print()
    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. 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]`

6. Convert the following **binary numbers to decimal and hexadecimal**.
   - **a.** `0011 0011`
   - **b.** `1011 1110`
   - **c.** `1111 0000`

7. Convert the following **decimal numbers to binary and hexadecimal**.
   - a. 126
   - b. 83
   - c. 29

8. Convert the following **hexadecimal numbers to binary**.
   a. B0A452
   b. 9A03DE
   c. 621097

9. The code below has seven errors for you to find and correct.

```
# makeInitials() takes in a full name and returns the initials
# (for example: "Freeman A. Hrabowski" would become "F.A.H.")
# Input:          name;     a string
# Output:         initials; a string of initials
def makeInitials(name):

    # separate first/middle/last names
    nameList = name.strip()

    for i in range(len(nameList)):
        currName = nameList[i]
        # take the first letter from each name
        firstLetter = nameList[currName][0]

        # format and add to the current initials
        tempInitial = currName.upper()
        initials = tempInitial + "." + initials

def main():
    myName = input("Please enter a name: ")
    myInitials = makeInitials(myName)
    print("The initials for", myName, " are:", myInitials)

main()
```

10. CHALLENGE PROBLEM:
    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).

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

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

14. 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] ]
```

15. 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.)

16. CHALLENGE PROBLEM:
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).

17. <u>Study with friends</u>! 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.

18. 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) )
```

19. More debugging – the code below has eight errors.

```
# findMin() takes in a list and returns its minimum value
def findMin(myList):
    currMin = myList(0)

    for i in range(len(myList)):
        if currMin < myList[i]:
            currMin = i
        return currMin


def main():
    myList = []
    myMin = 0

    # create a list of 10 items
    while len(myList) <= 10:
        newNum = int(input("Please enter a number " + \
                                "for the list:")
        myList.append(newNum)

    findMin(myList)
    print("The minimum is:", myMin)
```

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

21. More debugging – the code below has six errors

```
# power2() calculates 2 raised to the power passed in
# Input:   exponent; an integer for the power
# Output:  2 raised to the power of exponent
def power2(exponent):
    # BASE CASE: anything to the zero is 1
    if exponent == 0:
        return 1
    # RECURSIVE CASE
    else:
        prevPower = power2(exponent)
        return 2 * exponent

def main():
    myExp = -1

    while myExp < 0:
        myExp = input(int("Please enter a positive integer: "))

    myPower = power2(exponent)
    print("2^" + myExp + " = " + myPower)
```

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

23. 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 <u>how they work</u> and their <u>run times</u>)
   c. Linear search and binary search (again, should know <u>how they work</u> and their <u>run times</u>)
   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 13, 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!