Problem Solving Methodology in IT (COMP1001)

Midterm test for Group 1 (Tue and Fri)

16 October 2018

Your name: _____ Your student ID: _____

Your program: _____

Please read the instructions below carefully.

1. Before the commencement of the test:
    a. Put away your phones, calculators, and other electronic devices.
    b. Put down your name, student ID and program name on the test paper
2. About the questions and submissions:
    a. Each question carries the same weight.
    b. Answer all five questions.
    c. Write down your answers to Q1 to Q3 in the space provided on this paper.
    d. Submit the .py files for Q4 and Q5 to Blackboard. Name your files by "your student ID"_Q4.py and "your student ID"_Q5.py, respectively. You should upload your .py file as soon as you finish one. There is no limit on the number of submissions.
3. During the test:
    a. You can use everything available in your PC and access everything under Blackboard and nothing else.
    b. You may use pencil/ball pen, eraser and ruler and nothing else.
    c. No communication with others in any form
    d. Bathroom policy: one at any time. Raise your hand and get permission before going.
    e. Raise your hand if you have any question about the test paper.
4. At the end of the test period:
    a. The test will end at 6:20pm. You must stop when the time is up.
    b. If you finish early, you may submit the written answers and shut down your PC. But you cannot leave the venue until 6:20pm. You may use your own notebook to work on other coursework.

1. [Weight = 1] Consider a binary representation of a decimal number in the form of $a_2a_1a_0.a_{-1}a_{-2}a_{-3}a_{-4}a_{-5}a_{-6}a_{-7}a_{-8}a_{-9}a_{-10}$. List 4 smallest decimal numbers that can be represented exactly by this binary representation. We consider only nonnegative numbers. Include a succinct explanation for your answer.

2. [Weight = 1] What does the pseudocode below generate? Include a succinct explanation for your answer.

```
for outer in [1,2] do
      for middle in [3,4,5] do
            print(outer,middle)
            for inner in [6,7] do
                  print(outer,middle,inner)
      print(outer,middle,inner)

Function print(x,y)
   Input:  two integers x and y
   Output: print x first, separated by a space, and then print
           y. After that the next print() will start from a new
           line.
Function print(x,y,z)
   Input:  three integers x, y, and z
   Output: print x first, separated by a space, and then print
           y, separated by a space, and lastly print z. After
           that the next print() will start from a new line.
```

3. [Weight = 1] Given a string representing a real number in binary (such as "1001.010101"), write pseudocode for converting the real number in binary to decimal. The string always contains the radix point. The function signature is given below.

```
Function real_bin2Dec(r)
     Input:  r is a string representing a real number in binary
             (e.g., "101.001").
     Output: return the decimal value of the binary number in r.
```

You could use the two functions below in your pseudocode, and you cannot define other functions. Do not use *range()*. Instead, use [0, 1,…, *n*] for *range(n+1)* and [*m, m+1,..., n*] for *range(m, n+1)*.

```
Function posRadixPt(r)
     Input:  r is a string representing a real number in binary
             (e.g., "101.001").
     Output: return the index of "." in r. For example,
             postRadixPt("101.001") returns 3.

Function len(r)
     Input:  r is a string.
     Output: return the length of r.
```

4. [Weight = 1] Write a Python function called *factors()* that will return all the factors of a positive integer in a list. For example, the factors of 6 are [1, 2, 3, 6] and that of 312 are [1, 2, 3, 4, 6, 8, 12, 13, 24, 26, 39, 52, 78, 104, 156, 312]. The function signature is given below. You could use any built-in function and string method but cannot import modules.

```
Function factors(n):
    Input: n is a positive integer.
    Output: return a list of all the factors of n.
```

Also include the following statements in your .py file.

```
print(factors(6))
print(factors(90))
print(factors(312))
print(factors(3120))
```

5. [Weight = 1] Write a Python function that compares two strings based on the lexicographical order. The results could be "<", ">", or "=". The function signature is given below. You could use any built-in function but not string methods. You also cannot import modules.

```
Function compareStrings(s1,s2)
   Input: two strings s1 and s2
   Output:
   - return ">" if s1 is lexicographically greater than s2.
   - return "<" if s1 is lexicographically less than s2.
   - return "=" if s1 is lexicographically equal to s2.
```

Include the statements below in your .py file.

```
print(compareStrings("alpha","omega"))
print(compareStrings("omega","alpha"))
print(compareStrings("alphaomega","alphaomega"))
print(compareStrings("","omega"))
print(compareStrings("Benjamen","Ben"))
print(compareStrings("",""))
```

END