UNIVERSITY OF LONDON

GOLDSMITHS COLLEGE

B.Sc. Examination 2016

COMPUTER SCIENCE

IS51021B Problem Solving For Computer Science

Duration: 2 ¼ hours

Date and time: May 2016

This paper is in two parts: part A and part B. You should answer ALL questions from part A and TWO questions from part B. Part A carries 40 marks, and each question from part B carries 30 marks. The marks for each part of a question are indicated at the end of the part in [.] brackets.

You are not allowed to use calculators during the exam.

There are 100 marks available on this paper.

THIS PAPER MUST NOT BE REMOVED FROM THE EXAMINATION ROOM

Part A: : You should attempt all of these three questions

1. What is the value of each of the following Python expressions?
   a. 5 **2
   b. 5//2
   c. 5/2
   d. 5 % 2
   e. 5 * 2.0

[5 marks]

2. What is the value of each variable, x, y, a, b, and c after the following sequence of commands are executed:

```
x = 2
y = 3
y = x
x += y
a = x == y
b = not(a)
c = b + 4
```

[5 marks]

3. Write a Python function, `addUpTo, t`hat has a non-negative integer n as input and has the effect of, if n is greater than 0, adding up all the numbers from 0 to n.

For example the effect of addUpTo(3) would be to print 6 because
0+1+2+3 = 6

[5 marks]

4. Consider the list aList = ["me",1,'you',2,5]
   What is the value of each of the following Python expressions?
   a. aList[0]
   b. len(aList)
   c. aList[1:3]
   d. aList[-1]
   e. aList [2:-2]

[5 marks]

5. Explain (with the help a diagram) each of the following concepts:
    a. Graph
    b. Weighted Graph
    c. Tree

[5 marks]

6.
    a. What is a ***Path*** in a weighted graph?
    b. What is meant by a ***shortest path*** between two nodes?
    c. Name an algorithm that tries to compute a shortest path in a weighted graph.
    d. Describe a real real-world problem in which you may want to find a shortest path

[5 marks]

7. Explain every word and symbol in the following code:

```
import turtle
tom = turtle.Turtle()
```

[5 marks]

8. Write a function that uses the turtle in question 7 to draw a square with sides of size 100. Your function should use a loop.

[5 marks]

Part B: You should attempt two of these three questions

I. Iteration and Newton-Raphson

a. Explain, using a simple example, like binary search for a number between 1 and 100, how you can solve a problem by successive guesses that converge (get nearer and nearer) to an answer.

[4 marks]

b. Consider binary search: how would it be different if you were looking for a discrete quantity (like a whole number) or a continuous one (like a real number)

[2 marks]

We wish to use the Newton-Raphson method to find out where the function f meets the x axis, that is when f(x) is 0. Newton-Raphson works by computing successive guesses using the following scheme: you compute the n+l st guess, g_{n+1}, by considering the tangent line to f at $g_{n,.}$ Where the tangent line intersects the x axis is your next guess for where f meets the x axis.

c. Draw a diagram that makes that process clear.

[5 marks]

d. What function would you put in for x to compute the square root of 5? That is, what function has the property of evaluating to 0 when the input is the square root of 5. What is the derivative of that function? Call that function ***sqr5(x)***

[3 marks]

e. What is the derivative of f(x) in d. Call it ***sqr5Prime(x)***

[2marks]

f. Write, in Python, a version of Newton-Raphson that takes two functions, ***f*** and ***fPrime*** , and two numbers ***start*** and ***eps*** as inputs and tries to find a value for which f is almost 0 (within eps of 0)

[6 marks]

g. How would you sue your answer to f to compute the square root of 5?

[2 marks]

h. Write, in Python, a Newton-Raphson algorithm that will take as inputs n, inp and return a number that is approximately an nth root of inp

[6 marks]

II. Graph Models

A document which is currently written in English is to be translated into six other European Union languages. The cost of translating a document varies, as it is harder to find translators for some languages. The costs, in euros, are shown below.

	D	**E**	**F**	**G**	**H**	**I**	**S**
Danish**(D)**	-	120	140	80	170	140	140
English**(E)**	120	-	70	80	130	130	110
French **(F)**	140	70	-	90	190	85	90
German **(G)**	80	80	90	-	110	100	100
Hungarian**(H)**	170	130	190	110	-	140	150
Italian **(I)**	140	130	85	100	140	-	60
Spanish **(S)**	140	110	90	100	150	60	-

a. Draw the weighted graph that encapsulates this information.

[7 marks]

b. Each translator can only translate one pair of languages. What is the least number of translators you need to make sure that every document can appear in every language.

[2 marks]

c. What is the structure you need to find to minimise the total cost of translations .

[2 marks]

d. Find that structure, naming your algorithm and showing your work.

[7 marks]

e. Name two algorithms for solving this and explain how they differ.

[4 marks]

f. Write, in Python, any algorithm for solving problems of this sort. *You can use the class definition from the appendix*

[8 marks]

III. Sorting

a. Given an ordinary telephone directory, explain why it is easier to find a phone number of a named person than to find the person who has a particular telephone number. Your answer should include how many comparisons, at worst, it would take in each case, if you had a telephone directory with 31 entries.

[6 marks]

b. We wish to sort the following list using merge sort:

[3,1,5,4,2,6,4,8]

Go through the whole process, showing every comparison that you have had to make.

[10 marks]

c. How many comparisons did you make? How many comparisons would you have made if you were sorting this using Bubblesort?

[4 marks]

d. Write in Python a function that MergeSorts a list.

[10 marks]

Appendix:

```
class Graph:
def __init__(self, size, edges = []):
    self.matrix = [[0]*size for x in range(size)]
    self.size = size
    for edge in edges:
      if len(edge) == 2:
        self.addEdge(edge[0], edge[1], 1)
      elif len(edge) == 3:
        self.addEdge(edge[0], edge[1], edge[2])

  def vertices(graph):
    return range(graph.size)

  def addEdge(self, start, end, weight):
    self.matrix[start][end] = weight
    self.matrix[end][start] = weight

  def weight(self, start, end):
    return self.matrix[start][end]

  def connected(self, start, end):
    return self.weight(start,end) != 0

  def neighbours(self, node):
    answer = []
    for v in self.vertices():
      if self.connected(node, v):
        answer.append(v)
    return(answer)
```