THE UNIVERSITY OF WESTERN AUSTRALIA

Achieve International Excellence

Computer Science and Software Engineering

SEMESTER 1, 2013 EXAMINATIONS

CITS1401
Problem Solving and Programming

FAMILY NAME: _____ GIVEN NAMES: _____

STUDENT ID: | | | | | | | | | SIGNATURE: _____

This Paper Contains: **14** pages **(including title page)**
Time allowed: **2 hours 10 minutes**

INSTRUCTIONS:
Answer all questions. The marks for the paper total 90.
Write your answers in the spaces provided on this question paper.
No other paper will be accepted for the submission of answers.
Do not write in this space.

				X	

PLEASE NOTE

Examination candidates may only bring authorised materials into the examination room. If a supervisor finds, during the examination, that you have unauthorised material, in whatever form, in the vicinity of your desk or on your person, whether in the examination room or the toilets or en route to/from the toilets, the matter will be reported to the head of school and disciplinary action will normally be taken against you. This action may result in your being deprived of any credit for this examination or even, in some cases, for the whole unit. This will apply regardless of whether the material has been used at the time it is found.

Therefore, any candidate who has brought any unauthorised material whatsoever into the examination room should declare it to the supervisor immediately. Candidates who are uncertain whether any material is authorised should ask the supervisor for clarification.

Supervisors Only – Student left at:

This page has been left intentionally blank

Q1. Arithmetic

A point on a plane can be represented by two numbers *px, py* denoting its Cartesian coordinates; a circle can be represented by three numbers *cx, cy, r*, denoting the coordinates of its centre, and its radius.

(a) Define the following function. **3 marks**

def circumference(cx, cy, r):
#circumference returns the circumference of the circle cx, cy, r
e.g. *circumference(4, –4, 4) = 25.1*.

(b) Define the following function. **3 marks**

def origin(cx, cy, r):
#origin returns True if the origin lies inside the circle cx,cy,r, otherwise it returns False
e.g. *origin(3, 4, 6) = True*, and *origin(3, 4, 4) = False*.

(c) Define the following function. **4 marks**

def biggest(px, py, cx, cy, r): # assume px, py is inside cx, cy, r
#biggest returns a tuple containing three numbers that represent the largest circle
#that is centred at px, py and that fits inside the circle cx, cy, r
e.g. *biggest(5, 5, 1, 2, 9) = (5, 5, 4)*.

Q2. Booleans and testing

(a) Define the following function. **4 marks**

def fits(b1, b2, x):
#fits takes two Booleans b1, b2 and a number x; it returns True
#if x is equal to the number of Trues, otherwise it returns False
e.g. *fits(False, True, 1) = True*, and *fits(True, True, 3) = False*.

(b) Define the following function. **6 marks**

def test_fits():
#test_fits returns True if fits is correct, otherwise it returns False

test_fits should perform at least eight well-chosen tests on *fits*.

Q3. List iteration

(a) Use iteration over lists to define the following function. **4 marks**

def zip(xs, ys):# assume len(xs) == len(ys)
#zip returns a merged list containing alternating elements from xs and ys
e.g. *zip([1, 2, 3], [4, 5, 6]) = [1, 4, 2, 5, 3, 6].*

(b) Use iteration over lists to define the following function. **6 marks**

def cumulative (xs): # assume xs is not empty
#cumulative returns a list containing the cumulative sums of the elements from xs
#i.e. the first element, the sum of the first two elements, the sum of the first three, etc.
e.g. *cumulative([9, 7, –2, 4, 0, –1]) = [9, 16, 14, 18, 18, 17].*

Q4. List comprehensions

(a) Use a list comprehension to define the following function. **4 marks**

def between(x, y): # assume x <= y
#between returns a list holding all numbers that are multiples of x and factors of y
e.g. *between(3, 18) = [3, 6, 9, 18]*.

(b) Use a list comprehension to define the following function. **6 marks**

def pairs(n): # assume n >= 0
#pairs returns a list holding all pairs (x, y) such that 0 ≤ x < y < n
e.g. *pairs(4) = [(0,1), (0,2), (0,3), (1,2), (1,3), (2,3)]*.

Q5. List iteration

A simple lossless algorithm for compressing a list of Booleans replaces each sequence of consecutive values with the length of that sequence. All lists are assumed to start with (0 or more) *Trues*.

So e.g. *[True, True, False, True, False, False, False]* is compressed to *[2, 1, 1, 3]*, and *[False, False, True]* is compressed to *[0, 2, 1]*.

Use iteration over lists to define the following function. **10 marks**

def uncompress (xs):
#uncompress returns the list of Booleans that was
#compressed into the list of numbers xs
e.g. *uncompress([2, 1, 1, 3]) = [True, True, False, True, False, False, False]*.

Q6. String processing

(a) Define the following function. **4 marks**

def palindrome(xs):
#palindrome returns True if xs is a palindrome (i.e. if xs reads the same
#forwards or backwards), otherwise it returns False
e.g. *palindrome("abba") = True*, *palindrome("aba") = True*, *palindrome("abca") = False*.

(b) Define the following function. **6 marks**

def loseprefix(xs):
#loseprefix returns xs without its leading character x and
#any immediately-following repetitions of x
e.g. *loseprefix("aardvark") = "rdvark"*, *loseprefix("help") = "elp"*, *loseprefix("pppp") = ""*.

Q7. Dictionaries

Define the following function. **10 marks**

def substrings(xs):
#substrings returns a dictionary that records all of the non-empty, contiguous
#substrings from xs and the number of occurrences of each
e.g. *substrings('abab') = {'a': 2, 'b': 2, 'ab': 2, 'ba': 1, 'aba': 1, 'bab': 1, 'abab': 1}*.

Q8. Reduction and analogy

(a) Describe the basic principles and efficient operation of the problem-solving technique “reduction and analogy”. **5 marks**

(b) Illustrate your answer to (a) with a problem that is amenable to this technique, and sketch a solution to this problem that uses the technique. **5 marks**

Q9. Divide-and-conquer

(a) Describe the basic principles and efficient operation of the problem-solving technique “divide-and-conquer”. **5 marks**

(b) Illustrate your answer to (a) with a problem that is amenable to this technique, and sketch a solution to this problem that uses the technique. **5 marks**

END OF QUESTIONS

Blank page provided for rough work only

Blank page provided for rough work only

Blank page provided for rough work only