THE UNIVERSITY OF WESTERN AUSTRALIA

Achieve International Excellence

Computer Science and Software Engineering

SEMESTER 1, 2012 EXAMINATIONS

CITS1401
Problem Solving and Programming

FAMILY NAME: _____ GIVEN NAMES: _____

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

This Paper Contains: **11** 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 as a pair of numbers *px, py* denoting its Cartesian coordinates; a rectangle whose sides are parallel to the axes can be represented as four numbers *x1, y1, x2, y2*, denoting the coordinates of its bottom-left and top-right corners.

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

def area(x1, y1, x2, y2):
#area returns the area of the rectangle x1, y1, x2, y2
e.g. *area(2, -2, 10, 10) = 96*.

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

def half(x1, y1, x2, y2):
#half returns a tuple of four numbers representing a rectangle that has half the
#height and width of the rectangle x1, y1, x2, y2 and is centred on the same point
e.g. *half(2, -2, 10, 10) = (4, 1, 8, 7)*.

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

def flip(x1, y1, x2, y2):
#flip returns a tuple of four numbers representing a rectangle that has the
#same bottom-left corner and the same area as the rectangle x1, y1, x2, y2,
#but with its width and height swapped
e.g. *flip(2, -2, 10, 10) = (2, -2, 14, 6)*.

Q2. Booleans and testing

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

def one(x, y, z):
#one takes three Booleans and returns True iff exactly one of them is True
e.g. *one(False, False, True) = True*, and *one(True, True, True) = False*.

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

def test_one():
#test_one returns True iff one is correct

test_one should perform all tests required to establish the correctness of *one*.

Q3. List iteration

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

def alternating(xs):
#alternating takes a list of Booleans and returns True
#iff the values on xs alternate True and False
e.g. *alternating([False, True, False]) = True*, *alternating([False]) = True*, and *alternating([True, False, False]) = False*.

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

def triangle(xs):
#triangle returns a list of lists holding the elements from the list xs,
#where each list on the result has one element more than its predecessor.
#Surplus elements on xs are discarded
e.g. *triangle([9, 7, 2, 4, 6, 1]) = [[9], [7, 2], [4, 6, 1]]*, and
triangle([9, 7, 2, 4, 6, 1, 0, 5]) = [[9], [7, 2], [4, 6, 1]].

Q4. List comprehensions

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

def fives(n): # assume n >= 1
#fives returns a list containing the numbers in 1.. n inclusive
#that contain at least one 5
e.g. *fives(100) contains 5, 35, and 52* (plus lots of other values!).

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

def diagonals(n): # assume n >= 0
#diagonals returns a square list of length n where
#each number 0..2n-2 runs down a Left-Right diagonal
e.g. *diagonals(4) = [[3, 4, 5, 6], [2, 3, 4, 5], [1, 2, 3, 4], [0, 1, 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 simple(xs):
#simple returns the compressed version of the list of Booleans xs
```

Q6. String processing

(a) Use slicing and the string method *len* to define the following function. **4 marks**

def third(xs): # assume len(xs) % 3 == 0
#third returns the middle third of xs
e.g. *third("Python") = "th"*.

(b) Use slicing and the string methods *find* and *rfind*
to define the following function. **6 marks**

def middle(x, s): # assume x occurs at least twice in s
#middle returns the portion of s between the first and last occurrences of x
e.g. *middle("n", "Lyndon wants money") = "don wants mo"*.

Q7. Recursion over integers

Rabbits are prolific breeders. *n* mature rabbits produce *n/3* offspring in each month, and each offspring takes only one month to reach maturity (i.e. rabbits less than one month old don't breed). Assume that we start with twelve mature rabbits and no immature rabbits, and that rabbits never die.

Use recursion to define the following function. **10 marks**

def rabbits(n): # assume n >= 0
#rabbits returns the numbers of mature and immature rabbits after n months
e.g. *rabbits(1) = (12, 4)*, and *rabbits(2) = (16, 4)*.

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. Backtracking

(a) Describe the basic principles and efficient operation of the problem-solving technique “backtracking”. **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 PAPER