THE UNIVERSITY OF WESTERN AUSTRALIA

Computer Science and Software Engineering

SEMESTER 1, 2016 EXAMINATIONS

CITS1401 and CITS4406
PROBLEM SOLVING AND PROGRAMING

FAMILY NAME:_____ GIVEN NAMES: _____

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

This Paper Contains: **14** pages **(including title page and pages for rough work at the end)**
Time allowed: **2:00 hours (including reading time)**

INSTRUCTIONS:

There are 9 questions in total. All questions have equal marks. Answer all questions.

Write your answers in the spaces provided under each question and return all sheets.

There are three extra pages provided at the end for rough work. Work out your answers before you write them neatly in the provided spaces. Return these extra sheets as well.

No other paper will be accepted for the submission of answers.

Additional material is not allowed. Calculators are not allowed.

Total marks = 90.

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 and Boolean

A rectangle on a plane is represented by two points, p1 (upper left corner) and p2 (lower right corner) such that the rectangle sides are parallel to the xy-axes. Each point is represented by two coordinates x and y e.g. p1 = [x1, y1], p2 = [x2, y2], click = [x3, y3]. X-axis increases to the right and x -axis increases downwards. Define the following functions

(a) def insideClick(p1, p2, click): **3 marks**
    # returns True if the point click is inside the rectangle defined by p1 and p2.
    # Otherwise returns False. # e.g. insideClick([0,0], [10,10], [5,5]) = True

(b) def squareArea(p1, p2): **3 marks**
    # returns the area of the largest square that fits inside the rectangle defined by p1, p2
    # e.g. area([0,0], [10,20]) = 100

(c) def smallRectangle(p1, p2): **4 marks**
    # returns a rectangle (p3, p4) which is centred inside the input rectangle (p1,p2)
    # and has half the side lengths # e.g. smallRectangle([0,0], [40,20]) = ([10,5], [30,15])

Q2. Boolean and testing: Define the following functions

(a) def inside(p1, p2, p3, r): **4 marks**
    # returns True if the circle centred at point p3 = [x, y] is fully inside the rectangle p1, p2.
    # See Q1 for definition of rectangle e.g. inside([50,50],[200,100],[150,70],10) = True

(b) def numTrues(B, n): **4 marks**
    # takes a list of Booleans B and returns True if and only if there are exactly
    # 'n' number of Trues in B e.g. numTrues([True, False, True], 2) = True.

(c) def two(b1, b2, b3, b4): **2 marks**
    # Takes four Booleans and returns True if and only if there are exactly two Trues
    # e.g. two(True, False, True, False) = True.

Q 3. String processing: Define the following functions

(a) def uniqueWords(sent): **6 marks**

```
    # takes a string “sent” and returns a list of unique words in the list followed by their
    # number of occurrences e.g. count(“if and only if and if”) = [‘if’, 3, ‘and’, 2, ‘only’, 1]
```

(b) def changeSent(sent): **4 marks**

```
    # removes the leading empty spaces of string “sent”, the last character and every
    # occurance of the last character. Returns the resulting string
    # e.g. changeSent(‘    hello there, do you see’) =  ‘hllo thr, do you s’
```

Q 4. List iteration: Define the following functions.

(a) def swap(xlist): **3 marks**
    # takes a list of tuples and swaps the elements of each one
    # e.g. swap([(1,'a'), (2, 'b'), (3, 'c')]) = [('a', 1), ('b', 2), ('c', 3)]

(b) def makeOdd(ylist): **3 marks**
    # takes a list of integers and removes all even numbers from it
    # e.g. makeOdd([1, 3, 4, 6, 7, 10]) = [4, 6, 10]

(c) def primes(N): **4 marks**
    # returns a list of the first N prime numbers e.g. primes(5) = [2,3,5,7,11]

Q 5. List comprehension: Use list comprehension to generate

(a) a list of all odd numbers up to and including N that are not divisible by 5 i.e. [1, 3, 7, 9, 11, 13, 17, ….] **3 marks**

(b) a list that contains all possible pairs (a, b) of two lists A and B where (a, b) are the elements of A and B respectively **4 marks**

(c) a list of integers up to and including N that are multiples of x and y **3 marks**

Q 6. Dictionaries and decision structures

(a) def wordFrequency(fileName): **6 marks**
takes a string specifying a full path to a file, reads the file and returns a dictionary
that contains every word of the file and its frequency. Ensure that words with
upper or lower case and those ending with a full stop and comma are treated the
same. Assume that there will be a space after every full stop and comma.

(b) def maxWord(wfrqD): **4 marks**
takes a wfrqD dictionary generated by “wordFrequency” in (a) and returns a
highest frequency word and its frequency. If there are draws, return any.

Q 7. Recursion **10 marks**

Define a recursive function fastExpo(a, n) that performs fast exponentiation
i.e. fastExpo(a, n) = a^n where “n” is an integer. Write comments in your code to highlight the important components of a recursive function. Comments carry 3 out of the total 10 marks for this question.

Q 8. Backtracking

(a) Describe the basic principles of the backtracking method of problem solving.
5 marks

(b) One example of a problem that can be efficiently solved by backtracking is the n queens problem. Give another example of a problem that can be efficiently solved with this technique and detail a solution to the problem. **5 marks**

Q 9. Abstraction and Generalization

(a) Describe the basic concept and principles of the problem solving techniques "abstraction and generalization". **5 marks**

(b) Describe the basic concept and principles of the problem solving technique "reduction and analogy". **5 marks**

END OF EXAM PAPER

BLANK SHEET FOR ROUGH WORK

BLANK SHEET FOR ROUGH WORK

BLANK SHEET FOR ROUGH WORK