PRINCE OF SONGKLA UNIVERSITY

FACULTY OF ENGINEERING

Department of Computer Engineering

Midterm Examination: Semester 1
Date: 15th October 2016
Subject Number: 241-303
Subject Title: Discrete Mathematics (OLD)
Lecturer: Aj. Andrew Davison

Academic Year: 2016-2017
Time: 9:00 – **11:00 (2 hours)**
Room: Robot

Exam Duration: 2 hours

This paper has 3 pages.

Authorized Materials:

- Writing instruments (e.g. pens, pencils).
- Books (e.g. dictionaries) and calculators are **not** permitted.

Instructions to Students:

- *Answer questions in English.* Perfect English is **not** required.
- Attempt all questions.
- Write your answers in an answer book.
- Start your answer to each question on a new page
- Clearly number your answers.
- Any unreadable parts will be considered wrong.
- When writing programs, use good layout, and short comments; marks will not be deducted for minor syntax errors.
- The marks for each part of a question are given in brackets (...).

Question 1 (20 minutes; 20 marks)

Use induction to show that each equation is true:

a) $\frac{1}{1*2}+\frac{1}{2*3}+...+\frac{1}{(n-1)n}=1-\frac{1}{n}$, *when* $n \geq 2$ (10)

b) $2^n > n^2+n$, *when* $n>4$ (10)

Question 2 (15 minutes; 15 marks)

Consider the following C function:

```
void foobar(int a, int d)
{
  int s = a;
  int t = 0;
  while (s >= d) {
    s = s-d;
    t++;
  }
  printf("t=%d; s=%d\n", t, s);
}
```

The loop invariant S(k) is $d*t_k + s_k = a$, where $t_k = k$ and $s_k = a - d*k$ are the values of `t` and `s` after k iterations of the loop. `a` and `d` are both positive integers.

a) Prove that the loop invariant is correct, by induction on k. (10)

b) Give two examples of the output produced when `foobar` is called with different arguments. (2)

c) Say in words what `foobar` does. (3)

Question 3 (35 minutes; 35 marks)

a) Write a *recursive* C function `largestElem()` that takes **only** a `LIST` argument as input, and returns the *largest* element in the list. Assume that the list contains only positive integers. If the list is empty, the function returns -1. (15)

b) Write an *iterative* C function (i.e. one using loops) which does the same task as in (a). Do **not** use recursion. (15)

c) Compare the functions of part (a) and (b), and say in words which is more *space* efficient. Explain your decision.

Hint: efficiency in this case means the amount of memory used to store data. Do **not** use big-oh notation. (5)

Question 4 is on the Next Page.

Question 4 (50 minutes; 50 marks)

a) Work out the worst case big-oh running time for the following *recursive* function. Show all your working. Explain what you are using as the size value. (30)

```
int rSearch(int min, int A[], int pos, int size)
{
  if (pos == size)
    return min;
  else {
    if (A[pos] < min)
      min = A[pos];
    return rSearch(min, A, pos+1, size);
  }
}
```

A typical call to `rSearch()` is:

```
rSearch(A[0], A, 1, size);
```

which searches for the smallest value in the array A of size `size`.

b) Rewrite `rSearch()` to use a loop (or loops) instead of recursion. Do not use global variables. (8)

c) Work out the worst case big-oh running time for the iterative version of `rSearch()` from part (b). Use a structure tree, and show all your working. (7)

d) Compare the big-oh values for parts (a) and (c). Explain in words what the comparison means. (5)

--- End of Examination ---