PRINCE OF SONGKLA UNIVERSITY

FACULTY OF ENGINEERING

Department of Computer Engineering

Midterm Examination: Semester 1 **Academic Year**: 2008-2009

Date: 31st July 2008 **Time**: 9:00 – **11:00 (2 hours)**

Subject Numbers: 240-304 **and** 241-303 **Room**: R300

Subject Title: Mathematics for Computer Engineering
and Discrete Mathematics

Lecturer: Aj. Andrew Davison

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

(25 minutes; 25 marks)

Use induction to show that each equation is true:

a) $1 + 3 + ... + (2n + 1) = (n+1)^2$, when $n >= 0$ (12)

b) $\sum_{i=1}^{n} i\,(i!) = (n+1)! - 1$, for all positive integers (13)

Question 2

(15 minutes; 15 marks)

Consider the following C function:

```
void foobar(int a, int d)
{
  int r = a;
  int q = 0;
  while (r >= d) {
    r = r-d;
    q = q+1;
  }
  printf("q=%d; r=%d\n", q, r);
}
```

The loop invariant S(k) is $d*q_k + r_k = a$, where $q_k = k$ and $r_k = a - d*k$ are the values of `q` and `r` 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 `smallestElem()` that examines a `LIST` argument, and returns the *smallest* element in the list. Assume that the list contains only positive integers. If the list is empty, the function returns -1. Do **not** use global variables. *Hint*: before calling `smallestElem()`, set the first list element as the initial smallest value. (15)

b) Write an *iterative* C function (i.e. one using loops) which does the same task as in (a). Do **not** use recursion or global variables. (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 (variables space) used to store data. (5)

Question 4 is on the next page.

Question 4 (45 minutes; 45 marks)

a) Work out the worst case big-oh running time for the following *recursive* function. Show all your working. (15)

```
void sort(int vals[], int n)
{
  if (n == 1)
    return;

  sort(vals, n-1);   // sort first n-1 values

  // now insert vals[n-1] into the correct position in vals[]
  int temp = vals[n-1];
  int i = n-1;
  while ((i > 0) && (vals[i-1] > temp)) {
    vals[i] = vals[i-1];
    i--;
  }
  vals[i] = temp;
}
```

b) Rewrite `sort()` to use iteration (loops) instead of recursion. Do **not** use recursion or global variables. The new version should use the same input arguments as in part (a). (15)

c) Work out the worst case big-oh running time for the iterative version of `sort()` from part (b). Show all your working. (10)

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

--- End of Examination ---