PRINCE OF SONGKLA UNIVERSITY

FACULTY OF ENGINEERING

Department of Computer Engineering

Midterm Examination: Semester 1 **Academic Year**: 2006-2007

Date: Saturday July 29th, 2006 **Time**: 9:00 – **11:00 (2 hours)**

Subject Number: 240-304 **Room**: R 201

Subject Title: Mathematics for Computer Engineering

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 (20 minutes; 20 marks)

Use induction to show that each equation is true:

a) $\sum_{i=1}^{n} \frac{1}{i(i+1)} = \frac{n}{n+1}$, *when* n > 0 (10)

b) $1 + 3 + 5 + \ldots + (2n\text{-}1) = n^2$, *when* $n > 0$ (10)

Question 2 (15 minutes; 15 marks)

Consider the following C fragment:

```
scanf("%d", &n);
power = 1; i = 1;
while(i <= n) {
  power = power * x;
  i++;
}
```

The loop invariant S(k) is $\text{power}_k = x^{(ik)-1}$, where power_k and i_k are the values of `power` and `i` after k iterations of the loop.

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

b) What is the value of `power` after the loop terminates? Explain your answer. (5)

Question 3 (35 minutes; 35 marks)

a) Write a *recursive* C function `averageElem(s)` that takes **only** a `LIST` argument as input, and returns the *average* of all the elements in the list. If the list is empty, the function returns 0. Do not use global variables. `averageElems()` may call other functions. (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 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. (30)

```
void sort(int A[], int n)
{
  int imin, i;
  if (n > 1) {
    imin = 0;
    for (i=1; i < n; i++)
      if (A[i] < A[imin])
        imin = i;
    swap(A, n-1, imin);
    sort(A, n-1);
  }
}
```

Note: do **not** implement `swap()`. Assume that `swap()` has a constant running time.

b) Rewrite `sort()` to use loops instead of recursion. Do **not** use global variables. The new version should use the same input arguments as in part (a). Do not implement `swap()`. (8)

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