

PRINCE OF SONGKLA UNIVERSITY
FACULTY OF ENGINEERING
Department of Computer Engineering

Midterm Examination: Semester 1

Academic Year: 2005-2006

Date: 30th July 2005

Time: 9:00 – 11:00 (2 hours)

Subject Number: 240-304

Room: R200

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:

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

$$b) \quad 2^n > n^2 + n \quad , \text{when } n > 4 \quad (10)$$

Question 2

(15 minutes; 15 marks)

Consider the following C function:

```
void wold(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.

- Prove that the loop invariant is correct, by induction on k . (10)
- Give two examples of the output produced when `wold` is called with different arguments. (2)
- Say in words what `wold` does. (3)

Question 3

(35 minutes; 35 marks)

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

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

A typical call to `minFinder()` is:

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

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

- b) Rewrite `minFinder()` 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 `minFinder()` 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* ---