# PRINCE OF SONGKLA UNIVERSITY

## FACULTY OF ENGINEERING

## Department of Computer Engineering

| | |
|---|---|
| **Midterm Examination**: Semester 1 | **Academic Year**: 2003-2004 |
| **Date**: 27th July 2003 | **Time**: 9.00 – **11.00** (**2 hours**) |
| **Subject Number**: 240-311 | **Room**: R 200 |

**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 (…).

1. Use induction to show that each equation is true:

a) $n! >= 2^n$, when n >= 4 (10)

b) 1 + 4 + 7 + ... + (3n – 2) = n(3n – 1)/2, when n >= 1 (15)

(25 minutes; 25 marks

2. Consider the following C fragment:

```
scanf("%d", &n);
sum = 0;
for (i = 1; i <= n; i++)
  sum = sum + i;
```

The loop invariant S(k) is $\texttt{sum}_k = i_k(i_k - 1)/2$, where $\text{sum}_k$ and $i_k$ are the values of `sum` and `i` after k iterations of the loop. Assume that `n` is a positive integer.

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

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

(15 minutes; 15 marks)

3) a) Write a *recursive* C function `smallestElem()` that takes **only** a `LIST` argument as input, and returns the *smallest* element in the list. Assume that the list contains only positive integers with values less than 2000. If the list is empty, the function returns 2000. (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. (5)

(35 minutes; 35 marks)

**Question 4 on next page.**

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

```
void insertionSort(int s[], int n)
{
  if (n == 1)
    return;

  insertionSort(s, n-1);   // sort first n-1 elems

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

b) Rewrite `insertionSort()` to use iteration (loops) instead of recursion. Do not use recursion. 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 `insertionSort()` 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)

(45 minutes; 45 marks)

--- *End of Examination* ---