PRINCE OF SONGKLA UNIVERSITY

FACULTY OF ENGINEERING

Department of Computer Engineering

Midterm Examination: Semester 1 **Academic Year**: 2004-2005

Date: 31st July 2004 **Time**: 9.00 – **11.00 (2 hours)**

Subject Number: 240-304 **Room**: The Robot's Head

Subject Title: Mathematics for Computer Engineering

Lecturer: Aj. Andrew Davison

Exam Duration: 2 hours

This paper has 4 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 (25 minutes; 25 marks)

Consider the following C function:

```
int summer(int n)
{
  int db = 0;
  int i = 0;
  while(i < n) {
    db = db + n;
    i++;
  }
  return db;
}
```

The loop invariant S(k) is $db_k = k * n$ and $i_k = k$, where db and i_k are the values of db and i after k iterations of the loop.

a) Write down a precondition **and** a postcondition for summer(). (8)

b) Prove that the loop invariant is correct, by induction on k. (12)

c) What is the value of db after the loop terminates? Explain your answer. (5)

Question 3 is on the Next Page

Question 3 (40 minutes; 40 marks)

A *Sierpinski Gasket* is a fractal shape based around triangles.

In the first step of a gasket's creation, a triangle is drawn at the point (x,y) with a specified width and height (see Figure 1 below). Then three smaller triangles are drawn at its corners. The smaller triangles are width/2 wide and height/2 high. They are labelled as (2), (3), and (4) in the figure.

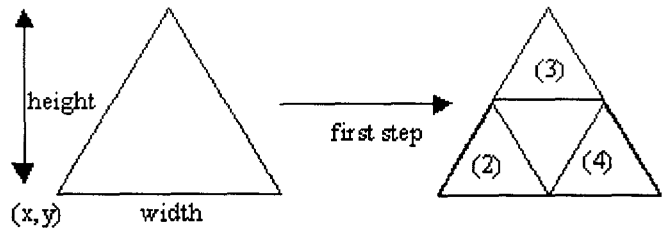

Figure 1. First Recursive Step in Generating a Sierpinski Gasket.

The next step is to repeat the process on the three smaller triangles, producing the triangles shown in Figure 2.

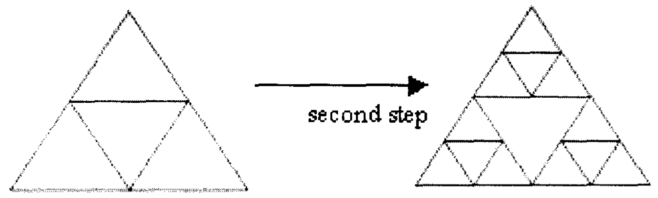

Figure 2. Second Step in generating a Sierpinski Gasket.

The recursive drawing continues by drawing smaller triangles inside each triangle.

Assume that you have two drawing functions available in C:

```
void setPen(double x, double y);
void drawLine(double xDistance, double yDistance);
```

setPen() moves the drawing pen to (x,y) without drawing anything. drawLine() draws a line from the current pen position, moving the pen xDistance in the x- direction, yDistance in the y- direction. The x- axis is across the screen, the y-axis is straight up.

a) Write a C function:

```
void drawTriangle(double x, double y,
                                double width, double height);
```

It draws a triangle at the point (x,y) with the specified width and height. Use setPen() and drawLine() to implement the function.
Do **not** implement setPen() or drawLine(). (10)

b) Write the C function:

```
void sierpinski(double x, double y, double width, double height);
```

It uses **recursion** to generate the triangles making up the Sierpinski Gasket. It uses drawTriangle() from part(a) to draw each triangle.

sierpinski() does not draw a triangle if its width or height is less than 0.2 units. (20)

c) Draw a *simple* diagram showing how sierpinski() executes when it is drawing the first two steps in the Sierpinski Gasket. The diagram should include the memory used by each function call. Explain the diagram in words. (10)

Question 4 (30 minutes; 30 marks)

The i th prefix average of a data[] array is the average of the first (i+1) elements of data[]. The average is stored in the i th cell of the prefixAvgs[] array:

prefixAvgs[i] = (data[0] + data[1] + ... + data[i]) / (i+1);

A C function which fills in the prefixAvgs[] array:

```
void prefixAverages(int data[], int n, double prefixAvgs[])
{
  double sum;
  int i, j;

  for(i=0; i < n; i++) {
    sum = data[0];
    for(j=1; j <= i; j++)
      sum += data[j];
    prefixAvgs[i] = ((double) sum) / (i+1);
  }
}
```

a) Work out the worst case big-oh running time for prefixAverages(). Show all your working. (10)

b) Rewrite the prefixAverages() function to have a **linear** worse case big-oh running time (i.e. O(n)). Write down the function **and** calculate the worst case big-oh running time. Show all your working. (15)

c) Explain in words why the code in part (a) is slower than the version you wrote in part (b). You should refer to the parts of prefixAverages() which cause the slowness. (5)

--- *End of Examination* ---