PRINCE OF SONGKLA UNIVERSITY

FACULTY OF ENGINEERING

Department of Computer Engineering

Midterm Examination: Semester 1 **Academic Year**: 2009-2010

Date: 30th July, 2009 **Time**: 9:00 – **11:00 (2 hours)**

Subject Number: 241-303 (240-304) **Room**: R200

Subject Title: Discrete Mathematics

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 (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 $power_k = x^{(ik)\,-1}$, 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 (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.

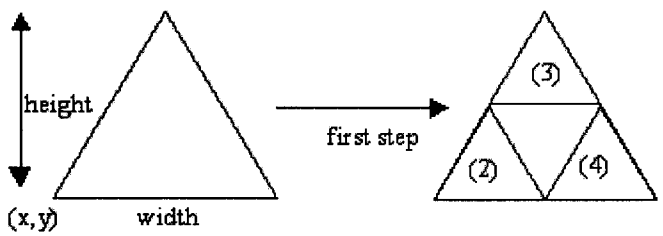

Figure 1. First Recursive Step in Generating a Sierpinski Gasket.

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

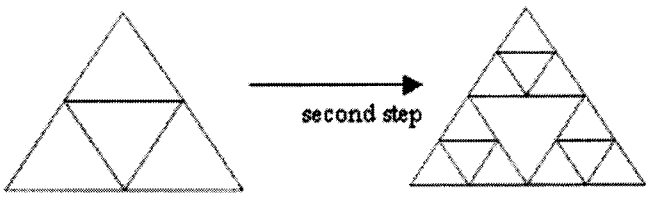

Figure 2. Second Step in generating a Sierpinski Gasket.

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 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. (25)

```
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 ---