School of Computing and Information Systems

comp10002 Foundations of Algorithms

Sample Mid-Semester Test #1

Student Number:

Time Allowed: Thirty-five minutes.

Authorised Materials: None.

Instructions to Students: This paper counts for 10% of your final grade. All questions should be answered in the spaces provided on the test paper.

And yes, there will be more lines in each set of boxes in the real test.

Question 1 (7 marks).

The *root mean square distance* between two sequences $X = \langle x_1 \cdots x_n \rangle$ and $Y = \langle y_1 \cdots y_n \rangle$ each containing n values is given by

$$\text{RMS}(X, Y) = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (x_i - y_i)^2}$$

Write a function

```
int calc_rms(double X[], double Y[], int n, double *rms)
```

that calculates the value $\text{RMS}(X, Y)$ for the arguments `X` and `Y`, where the sequences are stored in arrays following the usual C approach, and the argument `n` indicates the number of values in `X` and `Y` that are valid.

Your function should handle two distinct cases:

- if `n` is negative or zero, the return value of the function should be the constant `RMS_INVALID`, and the variable indicated by the pointer `rms` should not be changed in any way;
- otherwise, the return value of the function should be the constant `RMS_VALID`, and the computed RMS value should be assigned to the variable indicated by the pointer `rms`.

Note that you do not need to define the integer constants `RMS_INVALID` and `RMS_VALID`, and you do not need to know their values. Just assume that they are predefined and available for use in your function. You may make use of functions in `math.h` if you wish.

```
#include <math.h>

int calc_rms(double X[], double Y[], int n, double *rms)
{
    int i;
    double sum;
    if (n < 0 || n == 0) {
        return RMS_INVALID;
    } else {
        for (i=0; i<n; i++) {
            sum += (X[i] - Y[i]) * (X[i] - Y[i]);
        }
        *rms = sum/n;
    }
    return RMS_VALID;
}
```

(Yes, you'll be given more lines than this in the actual test paper.)

Question 2 (3 marks).

The following functions describe the computation behavior of a number of algorithms:

(a) $f_1(n) = 2n + 5$

(b) $f_2(n) = 2n - \log n + 1$

(c) $f_3(n) = f_1(n) + f_2(n)$

(d) $f_4(n) = f_1(n) \times f_2(n)$

(e) $f_5(n) = f_1(n) - f_2(n)$

(f) $f_6(n) = f_1(n)/f_2(n)$

Using the "big-Oh" notation, what is the *best description* of the asymptotic growth rate of each of the functions $f_1()$ to $f_6()$?

Function $f_1()$ is best described as being a member of the set:

f(n) = 2n+5 $O(n)$

Function $f_2()$ is best described as being a member of the set:

$O(n)$

Function $f_3()$ is best described as being a member of the set:

$O(n)$

Function $f_4()$ is best described as being a member of the set:

$O(n^2)$

Function $f_5()$ is best described as being a member of the set:

$O(n)$

Function $f_6()$ is best described as being a member of the set:

$O(1)$