Score: / 29 CIS 263

Name ⟨*Last, First*⟩ 2011:October:12(Wed)

Quiz #1

- During this test, you may not use any auxiliary materials or computational devices; i.e. this test is 'closed-book' (and 'closed-notes' etc.). But if you can't remember some little detail, ask the instructor.

[Acknowledgment: Some of these exercises are derived from Weiss.]

A. For each of the following program-fragments, give a worst-case analysis of the running time, $O(\ldots)$, of the program-fragment. For example, consider the following program-fragment:

```
count = 0;
for( i = 0; i < n; i++ )
    for( j = 0; j < i; j++ )
            count++;
```

For that program-fragment, the answer would be $\boxed{O(n^2)}$. (Partial credit may be awarded to excessive overestimates; e.g. it's technically correct to say that that code's running-time is $O(n^5)$, but n^5 is an excessive overestimate.) Show any intermediate steps you need to do to obtain your answers. As demonstrated with the example above, draw a box around each of your final answers.

1. [1 point]

```
count = 0;
for( int i = 0; i <= n; i += 2 )
    count++;
```

2. [3 points]

```
count = 0;
for( i = 0; i < n * n; i++ )
    for( j = 0; j < i; j++ )
            count++;
```

3. [3 points]

```
int count = 0;
double coefficients[] = { 1.2, -3.4, 5.6, ... };    // n values
double x = ...;     // some value
double poly_x = 0;
for ( int i = 0; i < n; i++ ) {
    double x_i = 1;
    for ( j = 0; j < i; j++ ) {
        x_i *= x;
        count++;
        }
    poly_x += coefficients[i] * x_i;
    count++;
    }
```

4. [2 points]

```
int count = 0;
double coefficients[] = { 1.2, -3.4, 5.6, ... };    // n values
double x = ...;     // some value
double poly_x = 0;
for ( int i = 0; i < n; i++ ) {
    poly_x = coefficients[i] + x_i * poly_x;
    count++;
    }
```

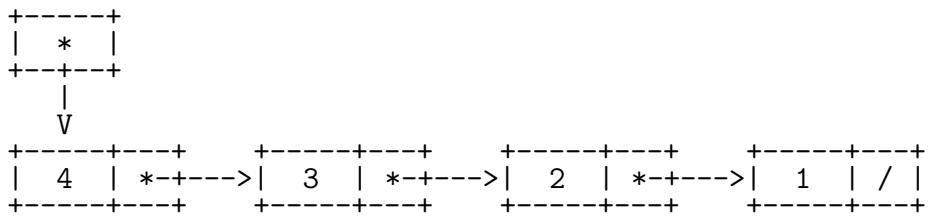
B. [6 points] Suppose the definition of `ListNode` is as follows:

```
struct ListNode {
    int element;     // The data in the node
    ListNode * rest;    // next node i.e. rest of list
    ListNode(int e, ListNode * r) : element(e), rest(r) { }
  };
```

Write a function `gen()` taking one argument say `n` of type `int`, returning a newly created list of `n` nodes containing the values from `n` down to `1` (or returning `NULL` if `n` is less than `1`). For example, suppose `head` is declared as follows:

```
ListNode * head;
```

Then the invocation `head = gen(4);` should set head to the following list:



Don't worry about whether some class may contain the function `gen()`; just write it here as an independent function:

C. [8 points] Write a recursive function to find the smallest value in an array of double-precision floating point numbers. Use **two recursive calls**: one call operates on the lower half of the array and the other on the upper half. The method returns **the index** of the smallest value, not the smallest value. If more than one such element are found in the array, return the lowest index.

```
int find_smallest(double x[], int L, int R) { ... }
```

the two parameters L and R specify the indices of the leftmost and rightmost elements (inclusive). So, for an array `myData[]` of N elements, the initial method invocation is as follows:

```
int idx = find_smallest(myData,  0,   N - 1);
cout << "The smallest element is at index " << idx << ".\n";
```

For example, `find_smallest({123.12, 105.04, -30.1, -10.0, 84.4, -13.12, 94.4}, 0, 6)` returns 2. Avoid creating a copy of the array to solve the problem. Notice that the method has an integer return value. Your recursive solution should use this return value properly.

D. [6 points] Write a complete C++ program that reads and stores an n-by-n matrix of integers. In this quiz, we don't have time to actually do anything with the data (such as matrix multiplcation or encryption or anything); but please still do this storing here anyway.
By the way, the number n will be the first integer to read; and it is guaranteed that n will be between 1 and 100. For example, the input might be as follows:

```
3
9 0 1
4 8 2
3 3 7
```