Name: last, first

CIS 263
2010:October:22(Fri)

Midterm Examination

- During this examination/quiz, you may not use any auxiliary materials or computational devices; i.e. this examination/quiz is ‘closed-book’ (and ‘closed-notes’ etc.). But if you can’t remember some little detail, ask the instructor.
- If your handwriting really is poorly legible, up to one point may be subtracted from your score.

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

A. [1 point] What does “$O()$” indicate? I.e. if $f(N)$ and $g(N)$ are functions, then which of the following is the expression “$f(N) = O(g(N))$” supposed to generally indicate? Circle your choice:

$f(N) < g(N)$ $\quad$ $f(N) \leq g(N)$ $\quad$ $f(N) = g(N)$ $\quad$ $f(N) \geq g(N)$ $\quad$ $f(N) > g(N)$

B. 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++;
    }
```

Score (Ex. B): / 9

C. [2 points] Order the following functions by growth rate: N, N^2, 2^N, $N * \lg(N)$, N^3

D. [2 points] Suppose an algorithm takes 1 second for input size 32. Then how large a problem can be solved in approximately one minute — say, 64 seconds — if the running time is linear (assume low-order terms are negligible)?

E. 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:

```
head:
+-----+
|  *  |
+--+--+
   |
   V
+-----+---+    +-----+---+    +-----+---+    +-----+---+
|  4  | *-+--->|  3  | *-+--->|  2  | *-+--->|  1  | / |
+-----+---+    +-----+---+    +-----+---+    +-----+---+
```

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

Score (Ex. E): / 6

F. Consider the following binary tree:

```
                 M
          ______/ \______
         /               \
        J                 U
     __/ #             __/ \__
    /                 /       \
   G                 Q         Y
  / \               / \       / \
 /   \             /   \     /   \
C     H           N     S   V     Z
/ \   # #         # \   # # # \   # #
A   E                P         W
# # # #             # #       # #
```

1. Give the preorder listing of the values in this binary tree:

2. Give the inorder listing of the values in this binary tree:

3. Give the postorder listing of the values in this binary tree:

4. List the leaves in this binary tree:

5. Draw the entire tree resulting from deletion of `N` from that binary tree::