ECE 1125 Data Structures and Algorithms – Fall 2015 – Exam 1 Solution

1. (a) Rank of complexities from least complex to most complex:

Θ(1), Θ(log(N)), Θ(N), Θ(N*log(N)), Θ(N^2), Θ(N^3), Θ(N!).

1. (b) Operations on a stack (assumes top starts at 0):

```
pop (returns error, top = 0)
pop (returns error, top = 0)
push 45 (returns success, top = 1, A = [45])
push 67 (returns success, top = 2, A = [45, 67])
pop (returns success, copies out 67, top = 1, A = [45])
pop (returns success, copies out 45, top = 0, A = [])
pop (returns error)
push 88 (returns success, top = 1, A = [88])
push 11 (returns success, top = 2, A = [88, 11])
pop (returns success, copies out 11, top = 1, A = [88])
```

1. (c) Operations on a queue of size 5:

```
Initial state: {h=0, t=0, A[X, X, X, X, X]}
deq (returns error, state unchanged)
deq (returns error, state unchanged)
enq 5 (returns success, st: {h=0, t=1, A[5, X, X, X, X]})
enq 4 (returns success, st: {h=0, t=2, A[5, 4, X, X, X]})
enq 3 (returns success, st: {h=0, t=3, A[5, 4, 3, X, X]})
deq (returns success, copies out 5, st:{h=1, t=3, A[5, 4, 3, X, X]})
enq 2 (returns success, state: {h=1, t=4, A[5, 4, 3, 2, X]})
enq 1 (returns success, state: {h=1, t=0, A[5, 4, 3, 2, 1]})
deq (returns success, copies out 4, st:{h=2, t=0, A[5, 4, 3, 2, 1]})
deq (returns success, copies out 3, st:{h=3, t=0, A[5, 4, 3, 2, 1]})
enq 0 (returns success, state: {h=3, t=1, A[0, 4, 3, 2, 1]})
enq -1 (returns success, state: {h=3, t=2, A[0, -1, 3, 2, 1]})
```

2. (a) Declaration and instantiation of doubly-linked list to hold double-precision floating point numbers:

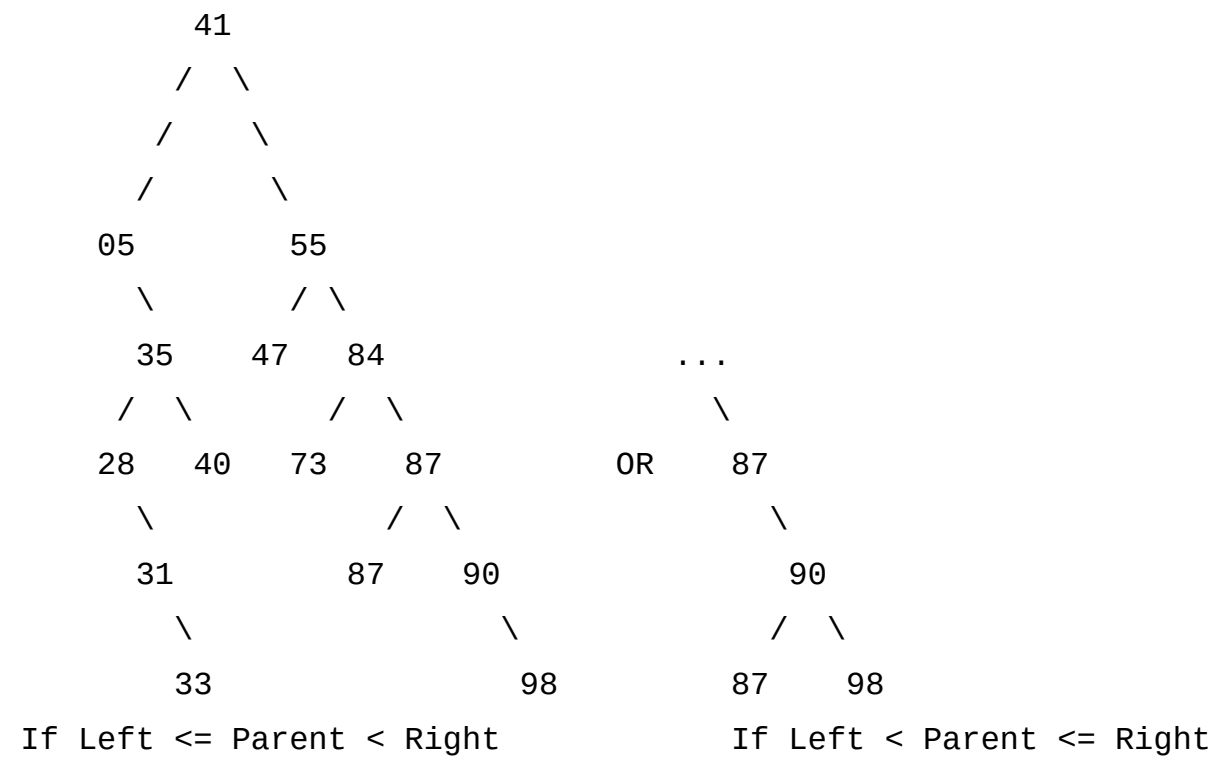
```
struct list_node {
	double data;
	struct list_node *prev;
	struct list_node *next;
};
struct list_node *head = NULL;
```

2. (b) Function in C:

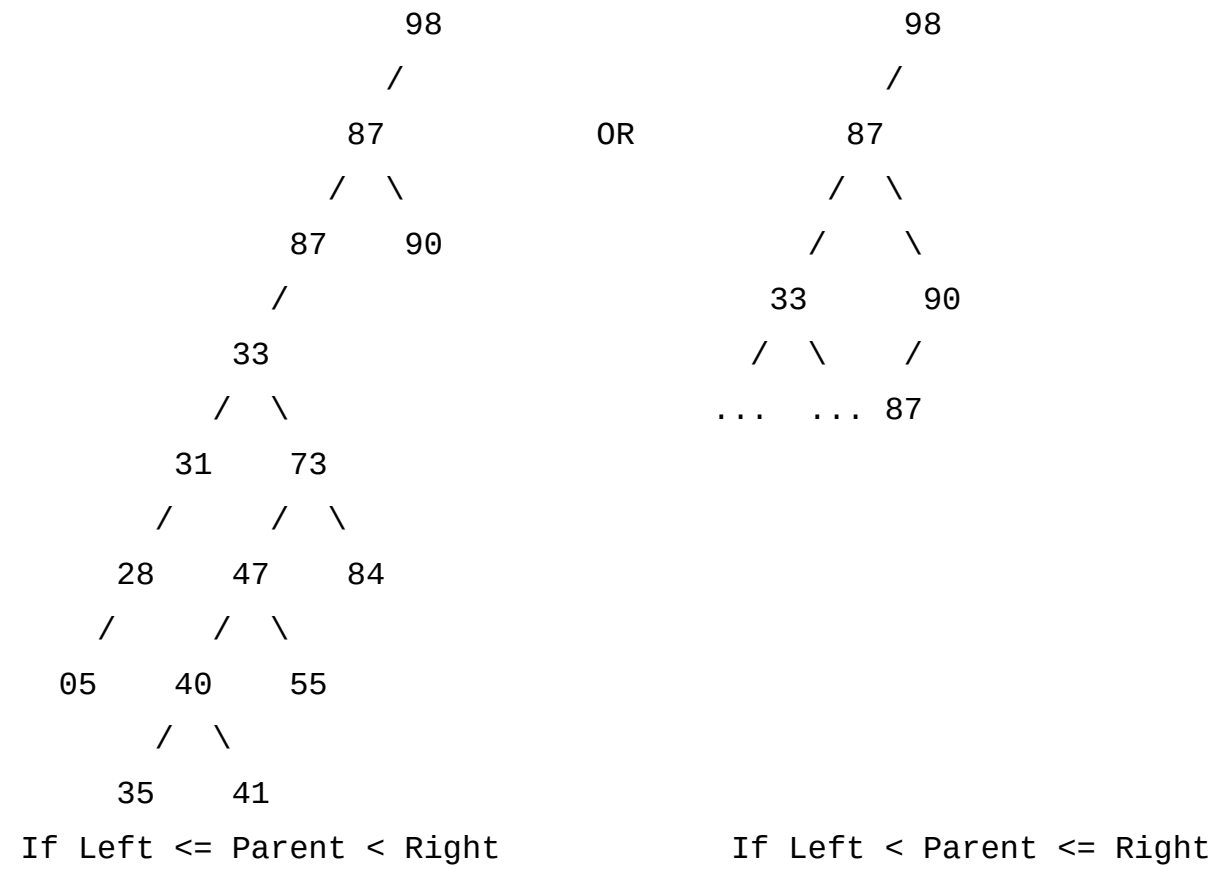
```
void print_reverse_below_mean(struct list_node *head)
{
	int count = 0;
	double mean = 0.0;
	struct list_node *n = head;
	while (n) {
		mean += n->data;
		count++;
		n = n->next;
	}

	if (count > 0) {
		mean /= count;
	}

	n = head;
	while (n && n->next) {
		n = n->next;
	}

	while (n) {
		if (n->data < mean) {
			printf("%f\n", n->data);
		}
		n = n->prev;
	}
}
```

2. (c) Time Complexity: Θ(N). Space Complexity: Θ(1).

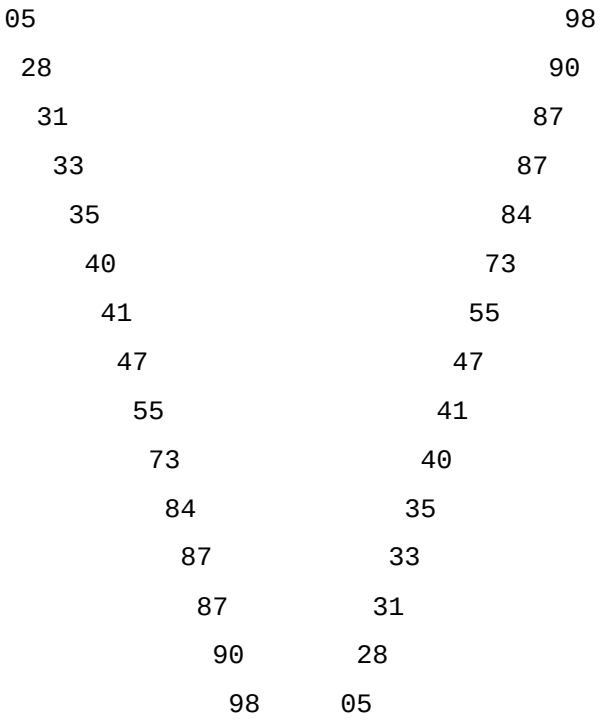
3. (a) Many solutions are possible. Binary search tree when inserted in the given order:



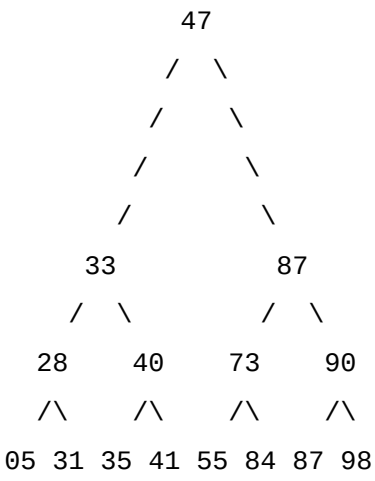
Next is shown inserting in reverse order:



3. (b) Two worst case binary search trees:



3. (c) Perfectly balanced tree (only Left < Parent <= Right is possible for perfect):



4. Recursive function for printing a BST in reverse order:

```
void print_reverse(struct tree_node *n)
{
      if (n) {
            print_reverse(n->right);
            printf("%d\n", n->data);
            print_reverse(n->left);
      }
}
```

5. (a) Approximate with 2^10 ~= 10^3 and solve for k:

2^k - 1 = 10^11

2^k ~= 10^11

log2(2^k) ~= log2(10^11)

k ~= log2((10^3)^3 * 100)

k ~= log2((2^10)^3 * 100)

k ~= 30 + log2(100)

The maximum depth must be an integer, so use the ceiling of this:

ceil(k) = 30 + 7 = 37

(b) The maximum number of comparisons to search is the maximum depth, 37.

(c) 10ns * 37 = 370 ns.

(d) 10ns * 10^11 comparisons = 10 * 10^(11-9) = 10^3 seconds.

6. (a) Time Complexity: Θ(sqrt(N)). Space Complexity: Θ(1).

(b) Each loop iteration performs 1 floating-point multiply, 1 floating-point comparison, and 1 floating-point add, so takes 4ns.

The loop runs until y*y == N, which occurs when y = sqrt(80000) = sqrt(8) * 100 ~= 283. Each whole number step takes 10^9 iterations of the loop, so the total time is:

283 x 10^9 x 4ns ~= 283 x 4 seconds = 1131 seconds. (Approximations are ok.)

(c) A faster binary-search based algorithm. (This is the basic idea in pseudo-code, this will not work for all inputs on a real machine without some modifcation.)

```
double mysqrt(double N)
{   double lo = 0.0, hi = N, mid;
    for (;;) {
        mid = (lo + hi) / 2.;
        if (mid * mid == N) {
            return mid;
        } else if (mid * mid < N) {
            lo = mid;
        } else {
            hi = mid;
        }
    }
}  /* Note: A real robust version would break if mid repeats itself. */
```

6. (d) Time Complexity: Θ(log2(N)). Space Complexity: Θ(1).