

Quiz 2 – 23 points possible

1. (5 pts) Give the Big-Oh notation for the following equations:

a)  $2n^3 - 6n + 30$

b)  $44n^{1.5} + 33n - 200$

c)  $5n^2 + 16n^2 \log n$

d)  $17n^2 \log n + 31n^3$

e)  $-3n^3 + 23n2^n$

2. (3 pts) Briefly define the following:

a) Black box design

b) Object oriented design

c) Abstract Data Type

3. (1.5 pts) Big-Oh notation is not the only notation used to express the runtimes of algorithms. List the names of 3 other relatives of the Big-Oh runtime notation.

4. (1.5 pts) By default, what happens when a thrown exception is never caught?

5. (4pts) Give the Big-Oh notation runtime for the following operations. If you believe there is more than one correct answer, you may justify your answer.

- a) Push back into a singly linked list with a tail pointer
- b) Push back into a doubly linked list with a tail pointer
- c) Dequeue from an array implemented queue (assume the most efficient implementation of the array)
- d) Remove from an unsorted array (do not include the search step)

6. (3 pts) The function enqueue into a queue is the same as push\_back onto a singly linked list with a head and tail pointer (basically insert the item in a new node at the end of the existing linked list). Write a function called enqueue that will take an item as a parameter and add it to the end of a linked list implemented queue. You may assume that the following function is a member of the queue class and the queue class is a friend class to the node class. You *must* use the following function prototype. **What is the Big-Oh running time of your function?**

```
// item is the item to enqueue into the queue (same as the push_back function on a
// linked list
void Queue::enqueue ( itemtype item ) {
```

7. (5 pts) Assuming a linked list with multiple occurrences of items, write a *recursive* function to return a pointer to the node with the *last* occurrence of an item in a singly linked list with only a head pointer. Return null if no such item exists. You may assume that the following function is a member of the list class and the list class is a friend class to the node class. You *must* use the following function prototype. **What is the Big-Oh running time of your function?**

```
// tmp is passed in as head the first time this function is called
// item is the item you are looking for.
Node* List::findLast ( Node* tmp, itemtype item ) {
```