

Name _____

This test is closed book, closed notes, and closed neighbor. Do your work on the paper provided. Answer on separate paper. You may answer questions in any order as long as you label the answers. Staple the pages together with this sheet on the front when you turn it in.

For the first two questions, write clearly and in complete sentences.

- (15 pts) Discuss the **three** different parameter passing modes in C++. Give examples and discuss when you use each.
- (5 pts) What does "don't dereference the null pointer" mean? Why is it bad? How do you avoid doing it?

- (10 pts) Write a complete C++ program that reads integers until end-of-file (or improper input, but the test is the same) and reports on how many positive integers were read. This problem does not need fancy data structures or functions.

```
template <typename T>
T mystery(const vector<T> & v) {
    if (v.size() <= 0) {
        return 0;
    }
    T value = v[0];
    for (unsigned i = 1; i < v.size(); i++) {
        if (v[i] < value) {
            value = v[i];
        }
    }
    return value;
}
```

- (18 pts) Consider the function at right. For 3 points each, answer the following questions concerning it.

- Given a vector **n** of int where
 $n[0] = 76$, $n[1] = 55$, $n[2] = 80$
 what is the value of `mystery(n)`?
- Given a vector **a** of string where
 $a[0] = \text{"horse"}$, $a[1] = \text{"panda"}$, $a[2] = \text{"fox"}$, and $a[3] = \text{"giraffe"}$,
 what is the value of `mystery(a)`?
- Give a concise yet descriptive name for this function.
- What parameter passing method was used for `v`? Why?
- What is the big-O computational complexity of this algorithm?
- Can we use this function with any type `T`? If not, what properties must type `T` have?

For all linked lists on this test, use the templated `NodeD<T>` class for a **doubly-linked list** which is **not circular** and which has member functions `getData`, `setData`, `getPrev`, `setPrev`, `getNext`, and `setNext`. Some questions refer to a specific type. Use that type as `T` and don't template your answer. But other questions need templated answers.

- (10 pts) Write a **mysteryLinked** function in C++ that has the same functionality as the **mystery** function in question 4 but operates on a doubly-linked list. Your function should be templated.
- (12 pts) Write a C++ function **removeEveryTarget** that takes a pointer to the first node in a doubly-linked linked list of integers as well as a target integer and removes all instances of the target integer from the list. It returns as its integer return value how many copies of target it removed from the list. The space should be returned to the heap. This function is not templated.
- (18 pts) We discussed the most common orders of complexity. List the following in order from fast to slow. Name an algorithm as an example for each. $O(N)$ $O(1)$ $O(N^2)$ $O(N \log N)$ $O(2^N)$ $O(\log N)$

- (12 pts) Show precisely what this code prints. Can you be precise? **Show your work.**

```
char a = 'y', b = '$', *p1 = &b, *p2 = new char('w'), *p3 = new char(*p1);
cout << a << b << *p1 << *p2 << *p3 << endl;
b = *p1;
delete p3;
p1 = p2;
cout << a << b << *p1 << *p2 << *p3 << endl;
```

Is there a memory leak in this code? Be specific in describing why or why not.