

Midterm Examination.

You are allowed to look at your notes, books, and the lecture slides. You are not allowed to search or roam the web. Write clearly. This exam is scheduled for 50 minutes. You then have 15 minutes to scan (if necessary) and upload your answers to CourSys as a PDF.

Question 1 (30 points total; 5 each)

Short answer questions. Complete sentences are not required. Justification for your answer is not required.

(a) Name two mechanisms for polymorphism in C++.

(b) What is wrong with the following C++ code?

```
void listSum() {  
    int* A = new int[10];  
    ...  
    sum = 0;  
    for(int i=0; i<10; i++) {  
        sum += A[i];  
    }  
    ...  
    delete A;  
}
```

(c) What is a *pure virtual function* in C++? What effect does it have on the containing class?

(d) What is the worst-case time complexity of *erase* for an array list implemented as a (growable) array?

(e) What is the worst-case time complexity of *erase* for an array list implemented as a linked list?

(f) Let the class BB be a subclass of the class AA, and the class CC be a subclass of BB. True or false? The following statement is legal C++:
AA* aa = new CC();

Question 2 (30 points total; 10 each)

- (a) What is the runtime behaviour of a thrown exception in C++?
- (b) What, in the context of this course, is delegation?
- (c) What is the *root* of a tree? What is special about it?

Question 3 (20 points)

Let *Node* be a class that represents a node in a rooted tree. It has a pointer to its *parent*, an integer *numChildren*, and an array of *numChildren* pointers to its *children*. The height of a node in the tree is the maximum distance (number of edges) to any of its descendants, and 0 if it is a leaf. Suppose each node has a member variable *height* which is initially set to infinity. Give pseudocode for an algorithm that (1) correctly computes all of the height values for the nodes in a tree, (2) returns the sum of all of the heights in the tree, and (3) does that all in one postorder traversal of the tree.

Question 4 (20 points)

Suppose that the only data type you have is a Deque ADT. (No arrays, no linked lists, no Stacks, etc.) Write pseudocode for a function that, given an even-sized Deque, returns a **perfect shuffle** of the Deque. A perfect shuffle is accomplished by splitting the items into the first half and second half, and then taking the first item from the first half, followed by the first item from the second half, followed by the second item from the first half, followed by the second item from the second half, etc., alternating between taking items from the first and second halves until all items are used. Since you only have Deques, you must return the shuffle as a Deque and use Deques as any intermediate structures you have.