CS200 Functional Data Structures
Solution to Ass 1

Beep Beep: Anas-am00634 and Affan-sa00310

March 1, 2016

Problem 1. Refresher

Please see the ass1.cpp file in the same folder.

Problem 2. Rotation

Following is C++ code of a function that Rotates a doubly-linked list by r.

```
template<class T>
T DLList<T>::rotate(int r) {
r = r % n;
Node* temp = getNode(r);
dummy.prev->next = dummy.next;
dummy.next->prev = dummy.prev;
dummy.next = temp;
dummy.prev = temp->prev;
}
```

The basic concept of our function is to change the position of dummy such that it lies before the rth index. Our complexity is just the complexity of O(1+min(r,n-r)) because getNode(r) has complexity O(1+min(r,n-r)) (It can traverse back if $r > n/2$). This function would work for any data type.

Problem 3. A Degenerate Tree

To have a height n-1, each element must form a single branch of the previous node. The tree should be a singly branched tree at each node. This is possible as at each node, there are two choices, either left or right child. Since there are n elements, and one element would form the root, there are n-1 different nodes possible. Therefore, the total combination of having a tree of height n-1 is 2^{n-1}.

Problem 4. Inverse?

Since the added object is necessarily a terminal branch, therefore yes, we would get the same tree. This is because the add function first searches through the tree to find the place for x and then adds x as node to the last found node (left or right depending on value). When remove is called, it would remove x easily because x is a leaf and the algorithm simply calls splice on x.