Jennifer Manning

1. Each of your responses should give bounds on the running time in terms N and M, where N is the number of players, and M is the number of passes. Justify each of your answers in paragraph form.

   a. The upper bound would be O(N^2). This is because the worst case would be M=N. In this case the first element to be deleted would be the Nth element, then after iterating through the list another n times, the next element would be deleted, and so on for N elements. So for each element, an iteration of n must occur. So that is N*N time.

   b. The upper bound would also be O(N^2). This is because it would have the same worst case listed above. As unlikely as it is, there is the possibility that a random number generator could generate the same random number each time, and for that random number to be N.

2. Write an algorithm, in pseudo code, to reverse a singly linked list in O(N) time. Justify your answer in paragraph form.

```
Node= List.Begin()
Prev=NULL
Next= Node->Next()
for(i=0; i<list.size(); i++)
        Node->Next= Prev
        Prev=Node
        Node=Next
        Next=Next->Next
```

This is O(N) because it will only loop through the link once for every object. Each node in the list will have its pointer to the next object (assuming it is a forward linked list), point to its prev object, which would reverse the order of the list. For example, the first object will point to Null, then the following object will now point to the previous. The “Next” node will store the node that has now been disconnected, and then on the next iteration, move on to the following.

3. Give an upper bound on the running time for each of the following operations, justify your answer in paragraph form.

   a.Deleting a key from a singly linked list is O(N), unless given the previous node, in which case it would be O(1). In other words, if they node to be deleted in the only argument given, then one must iterate through the list until that element is reached, the worst case of this would be deleting the last element, in which case there would be N elements gone through. However, if the previous node was also given as an argument, then it would be possible to say Prev->Next=Node->Next, and this would be O(1);

   b. Deleting a key from a doubly linked list would be O(1). This is because the previous node is able to be accessed by the given node. As disccused above, if that node is given, then it is fairly easy and quick to delete the unwanted node. There would be no iteration necessary, only code to change the pointers from the previous and following nodes to point to each other.

4. Let A and B be sets (a collection of unique items) expressed using linked lists. Provide pseudo code for computing A UNION B in O(1) time, where the result is the union of the two sets. Assume the sets are disjoint (have no elements in common). You may destroy A and B in the process.

Asumming both A and B are doubly linked lists:

```
A.End->Prev=B.Begin
B.Begin->Next=A.End
```

5. Repeat 4, but assume the sets are not disjoint. The union operation must then remove any duplicate items. What is the best running time you can accomplish (asymptotically) for this operation?

```
for(element1:A)
    for(element2:B)
        if element1==element2
            B.delete(element2)
    NewList.push_back(element1)
    A.delete(element1)
B.End->Prev=NewList.End
NewList.Begin->Next= B.End
```

This would be O(N^2), because it must iterate through all the elements of B for each element in A.