Name: _____

Write all of your responses on these exam pages. If you need extra space please use the backs of the pages.

1 Short Answer (7 Points Each)

1. Given the following function, what operations will need to be overloaded in the class T for this code to compile?

```
template <class T>
T square(T n)
{
  return n * n;
}
```

2. What is a container and what is an iterator?

3. What does LIFO mean and what data structure uses this type of access?

4. What does FIFO mean and what data structure uses this type of access?

5. Describe two operations that stacks must perform.

6. Describe two operations that queues must perform.

7. What are some of the advantages that linked lists have over arrays?

8. Write an implementation for the linked list version of the ListCollection class for the concatenation operator +. Recall that the nodes of the list are `ListNode` type and that the ListCollection class had the following functions.

   void setElement(int, T); T getElement(int); void clear(); int size(); int capacity(); void pushFront(T); void pushBack(T); T popFront(); T popBack(); void insertOrdred(T); void removeElement(T); void insert(int, T); void remove(int);

```
template <class T>
const ListCollection<T> ListCollection<T>::operator+(const ListCollection &right)
```

2 Coding (10 Points Each)

Given the following specification for the LinkedList class and ListNode class, write the implementations of all the functions.

```
1  using namespace std;
2
3  template <class T>
4  class ListNode
5  {
6    public:
7      T value;
8      ListNode<T> *next;
9
10     ListNode(T nodeValue)
11     {
12         value = nodeValue;
13         next = nullptr;
14     }
15 };
16
17 template <class T>
18 class LinkedList
19 {
20   private:
21     ListNode<T> *head;
22
23   public:
24     LinkedList()
25     {
26         head = nullptr;
27     }
28
29     ~LinkedList();
30     void appendNode(T);
31     void insertNode(T);
32     void deleteNode(T);
33     void displayList() const;
34 };
```

- `~LinkedList()` The destructor removes all elements from the list without memory leaks.
- `appendNode(T)` This function will append a node onto the end of the list.
- `insertNode(T)` This function will insert a node into the list so that if the list is currently ordered the new list will also be ordered.
- `deleteNode(T)` This function deletes the node that has the same value as the input parameter. If the input value is not in the list the original list is unaltered.
- `displayList() const` Writes the list contents to the screen.

---

1. `~LinkedList()`

2. `appendNode(T)`

3. `insertNode(T)`

4. `deleteNode(T)`

5. `displayList() const`