Lab 09
Arrays and Lists

Section 1: Guess program outputs.

1.

```
#include <iostream>
#include <string>
using namespace std;

template <class T>
class LinkedList
{
protected:
    struct ListNode
    {
       T value;
       ListNode *next;
       ListNode(T value1, ListNode *next1 = nullptr)
       {
          value = value1;
          next = next1;
       }
    };
    ListNode *head;
public:
    LinkedList() { head = nullptr;  } // Constructor
    ~LinkedList();                    // Destructor
    void add(T value);
    void remove(T value);
    void displayList() const;
};

template <class T>
void LinkedList<T>::add(T value)
{
   if (head == nullptr)
      head = new ListNode(value);
   else
     {
       // The list is not empty
       ListNode *nodePtr = head;
       while (nodePtr->next != nullptr)
          nodePtr = nodePtr->next;

       // Create a new node and put it after the last node
       nodePtr->next = new ListNode(value);
     }
}

template <class T>
void LinkedList<T>::remove(T value)
{
  ListNode *nodePtr, *previousNodePtr;

   // If the list is empty, do nothing
   if (!head)  return;
```

```
   // Determine if the first node is the one to delete
   if (head->value == value)
   {
     nodePtr = head;
     head = head->next;
     delete nodePtr;
   }
   else
   {
     nodePtr = head;

     while (nodePtr != nullptr && nodePtr->value != value)
     {
        previousNodePtr = nodePtr;
        nodePtr = nodePtr->next;
     }
     // Link the previous node to the node after nodePtr
     // then delete nodePtr
     if (nodePtr)
     {
        previousNodePtr->next = nodePtr->next;
        delete nodePtr;
     }
   }
}

template <class T>
void LinkedList<T>::displayList() const
{
   ListNode *nodePtr = head;
   while (nodePtr)
   {
      cout << nodePtr->value << "    ";
      nodePtr = nodePtr->next;
   }
}

template <class T>
LinkedList<T>::~LinkedList()
{
  ListNode *nodePtr = head;  // Start at head of list
  while (nodePtr != nullptr)
  {
      ListNode *garbage = nodePtr;
      nodePtr = nodePtr->next;
      delete garbage;
  }
}

int main()
{
      LinkedList<string> list;

      list.add("A");
      list.add("C");
      list.add("E");
      list.add("F");

      cout << "Here are the initial names:\n";
      list.displayList();
      cout << "\n\n";
```

```
      cout << "Now removing E.\n\n";
      list.remove("E");
      cout << "Here are the remaining elements.\n";
      list.displayList();
      cout << endl;

    return 0;
}
```

2.

```
#include <iostream>
#include <string>
#include <list>
using namespace std;

int main()
{
      list<int> myList;
      list<int>::iterator iter;

      for (int x = 0; x < 100; x += 10)
            myList.push_back(x);

      // Display the values
      for (iter = myList.begin(); iter != myList.end(); iter++)
            cout << *iter << " ";
      cout << endl;

      myList.reverse();

      // Display the values again
      for (iter = myList.begin(); iter != myList.end(); iter++)
            cout << *iter << " ";
      cout << endl;

      return 0;
}
```

Section 2: Review Questions and Exercises

1. Describe the two parts of a node.

2. What is a list head?

3. What signifies the end of a linked list?

4. What is a self-referential data structure?

Section 3: Programming Challenges

1. Simple Linked List Class
Using an appropriate definition of ListNode, design a simple linked list class with only two member functions, a default constructor and a destructor:
void add(double x);
boolean isMember(double x);
Linkedlist( );
~Linkedlist( );
The add function adds a new node containing x to the front (head) of the list, while the isMemberfunction tests to see if the list contains a node with the value x.
Test your linked list class by adding various numbers to the list and then testing for membership.

2. List Copy Constructor
Modify your list class of Programming Challenge 1 to add a copy constructor.
Test your class by making a copy of a list and then testing membership on the copy.

3. List Print
Modify the list class you created in the previous programming challenges to add a print member function. Test the class by starting with an empty list, adding some elements, and then printing the resulting list out.