CSCE A211 Exam #3 **Name _____**
Sample Exam #3 Questions

1. Short Answer. Provide a brief answer to the following questions.

a. Your C++ program has multiple classes that are structurally and algorithmically identical except for the data type that the classes operate on. What programming construct can you employ to eliminate the redundant code and have only one class?

b. On the threading lab why did we have to add a mutex to the function that ran in a thread to increment the counter?

c. Give the definition for a variable named **idMap** that is a C++ STL map template class that maps from an integer to a string.

d. You have a compiled C++ or Java program written a year ago. How is it possible for this program to call a function/method in new code that you wrote today without having to recompile the old code?

2. Find The Bugs

All of the following code snippets have at least one bug or problem. Assume the necessary headers/includes/imports are in place to make a working program. Identify the bug and state how you would correct the problem.

a. This program should fire off a thread to add two numbers, add 10 to the sum in main, then output the result.

```
int sum;

void func(int a, int b)
{
    sum = a+b;
}
```

```
int main()
{
    thread t(func, 2,4);
    int total = sum + 10;
    cout << total << endl;
}
```

b. Consider a file named `data.txt` that contains the following two names:

```
Myra Mains
Mark A. Place
```

Here is a program in the same folder that intends to output each name:

```
 string s;
 ifstream inData;

 inData.open("data.txt");
 inData >> s;
 cout << s;
 inData >> s;
 cout << s;
 inData.close();
```

3. Short Code

Give the output of the following code snippets. Assume the necessary headers/includes/imports are in place to make a working program.

a. C++ classes

```
class Foo
{
 public:
   void print1();
   virtual void print2();
};
void Foo::print1()
{
  cout << "foo1" << endl;
}
void Foo::print2()
{
  cout << "foo2" << endl;
}


class Bar : public Foo
{
 public:
   void print1();
   void print2();
};
void Bar::print1()
{
  cout << "bar1" << endl;
}
void Bar::print2()
{
  cout << "bar2" << endl;
}


int main()
{
  Bar b;
  Foo *p = &b;

  p->print1();
  p->print2();
}
```

b. Given the following template class:

```
template <class T>
class Foo
{
 private:
   T value;
 public:
   Foo(T data);
   void print();
};
```

Give the declaration for a variable named **x** of type Foo that sets the type of the template class to an int and initializes **value** to 10.

4. C++ Standard Template Library Classes

Listed below is the skeleton of a program that loops until the user enters a non-negative number. Complete the program so it uses either vector(s) or map(s) to output a histogram of the numbers entered (i.e. a count of how many of each number is entered). For example, if the user enters 3, 4, 4, 3, 4, 2, 99 then the output would be something like:

2 appears 1 times
3 appears 2 times
4 appears 3 times
99 appears 1 times

There should be no limitation on the numbers entered except that they should be non-negative and fit in an integer. Hint: it is easier to use a map.

```
int main()
{
 int num;
 do
 {
        cin >> num;









 } while (num >= 0);









 return 0;
}
```

5. Linked Data Structures

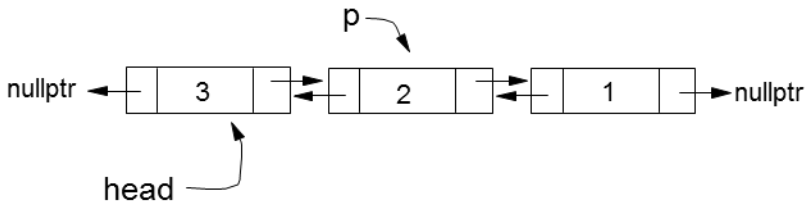
Here is code that creates a doubly-linked list. To simplify the exam, everything in the class is public, even though this is not a good programming practice!

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

class Node
{
 public:
  int data;
  Node *previous, *next;

  Node();
  Node(int data, Node *previousPtr, Node *nextPtr);
};
Node::Node() : previous(nullptr), next(nullptr)
{ }
Node::Node(int nodeData, Node *previousPtr, Node *nextPtr) :
      data(nodeData), previous(previousPtr), next(nextPtr)
{ }


int main()
{
 Node *head = nullptr, *temp = nullptr;

 for (int i=0; i < 4; i++)
 {
        if (head == nullptr)
                head = new Node(i, nullptr, nullptr);
        else
        {
                temp = new Node(i, nullptr, head);
                head->previous = temp;
                head = temp;
        }
 }
 /* In part b you will write code here
 */

 return 0;
}
```

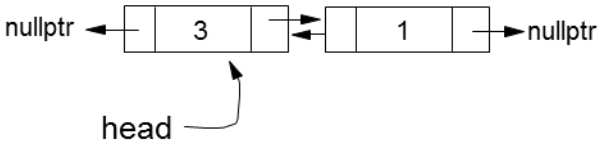
a) Draw the list structure (in the same style as the picture in part c, next page) that is created when the program runs. Make sure to show which element "head" is pointing to.

b) The program as it is written does not display the contents of the list. It also does not delete the nodes that are dynamically created. Write code that can execute at the end of the program that (1) displays the data variable of each list node, and also (2) deletes each node after it is printed.

c) You are given a variable "p" that points to some node in the linked list that you wish to delete. Write "Delete" code that will delete this node, but still maintains the integrity of the doubly linked list. For example, if the list appears as below:



Then after deleting the node pointed to by "p" we get the list:



Write code to delete the node pointed to by "p". You should take into account that "p" may point to any node in the linked list, including the first node or the last node in the list, and that there may be only one node in the list (in this case, "p" == "head"). Assume there is at least one element in the list.