Here is the final.

- To do the final, you can
    - print out the final, hand write it, and scan or photograph it
    - type a seperate text document with the answers
- There is a D2L dropbox for the written portion
- The instructions for the programming portion are at the end of the test.
- Please try to hand everything in by 10pm tonight
    - I'm not going to be a strict on this, I want everyone to hand it in and I know we might run into issues
- Please try to solve things in the easiest way possible, don't overthink it!
- I will be available for questions by
    - email: zachary.ankenman@pcc.edu
    - Google Meetings
        - https://meet.google.com/ocr-ugsx-eqf
        - Otherwise, to join by phone, dial +1 402-442-0168 and enter this PIN: 517 628 502#
    - Google Hangouts: zachary.ankenman@pcc.edu

Name____

1. Print out only the negative numbers of a linked list with Nodes as defined below using a loop and pointers.
   ```
   struct Node {
           double data;
           Node *next;
   };

   void printList(Node *n);
   ```

2. Print out only the negative numbers of a linked list using recursion. Same function signature and node definition as above.

3. Declare and define a class called Student.
    a. It has private member variables: char* name and ClassList* classList
    b. It has private static member variable: int numStudents
    c. It has a constructor with one const char* parameter.
        i. It allocates just enough memory to store the string parameter and copies the input string to name.
        ii. It allocates dynamic memory for classList using a no-argument constructor
        iii. It increments numStudents.
    d. It has a no argument destructor which decrements numStudents. What else does it need to do?
    e. It has const getter functions for name and classList, and a static const getter for numStudents.
    f. It has a setter function for name which allocates new memory and deletes the old name.
    g. It has an addClass function which has an int id parameter and calls this ClassList member function using the classList object: `void addClassById(int id);`
    h. It has a printToConsole function which prints name to the console and calls this ClassList member function: void printToConsole();

4. Recursively search a null terminated linked list for a data element equal to val. Return true if at least one matching element is found, false otherwise. The node structure is defined below along with the function signature.

```
struct Node {
      int data;
      Node *next;
}
bool searchList(Node *n, int val);
```

5. Inheritance: what prints out? Approximate the ordering when there is ambiguity, but note that not all ordering is ambiguous.

```
class Parent {
public:
      Parent() { cout << "Parent Constructor\n"; }
      ~Parent() { cout << "Parent Destructor\n"; }
      virtual void printStuff() { cout << "Parent printStuff\n"; }
};
class ChildA : public Parent {
public:
      ChildA() { cout << "ChildA Constructor\n"; }
      ~ChildA() { cout << "ChildA destructor\n"; }
      void printStuff() override { cout << "ChildA printStuff\n"; }
      void printStuff(int x){cout << "ChildA printStuff " << x <<
endl;}
};
class ChildB : public Parent {
public:
      ChildB() { cout << "ChildB Constructor\n"; }
      ~ChildB() { cout << "ChildB destructor\n"; }
};
int main() {
      ChildA a;
      a.printStuff(9);
      ChildB b;
      b.printStuff();
      return 0;
}
```

6. Circle and label the base cases and recursive calls. Calculate output for n == 4

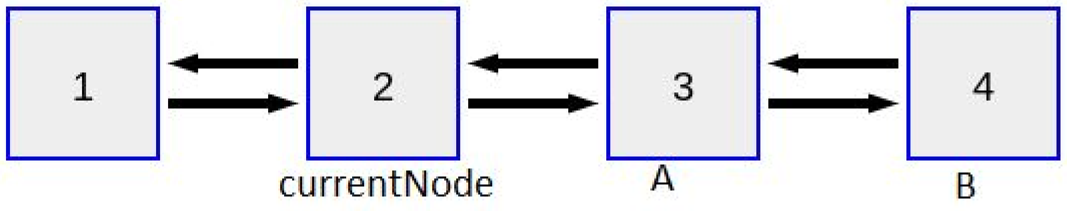
```
int recFunc(int n) {
	if (n == 0) return 2;
	if (n == 1) return 1;
	return recFunc(n - 1) + recFunc(n - 2);
}
```

7. If we had a linked list with the Node structure defined below, how would we access the Nodes labeled A & B based on the pointer to a Node labeled currentNode?

```
struct Node {
	double data;
	Node *next;
	Node *prev;
}
```

A: ______________________________________

B: ______________________________________



8. Assume we have dynamically allocated memory for an array x. Which would be the correct way to deallocate that memory?
    a. delete x;
    b. delete[] x;
    c. delete x[];
    d. delete x[SIZE];

9. Where is dynamically-allocated memory stored?

10. What would print out in this code snippet?

```
int x = 3;         // address of x is 0x100
int* y = &x;       // address of y is 0x108
int z = 4 * *y;    // address of z is 0x110
cout <<  x << endl;
cout << &x << endl;
cout <<  y << endl;
cout << *y << endl;
cout <<  z << endl;
```

11. What prints out?
```
class Thing {
      int data;
public:
      Thing(int newData) { data = newData; }
      void printData1(int data) { cout << data << endl; }
      void printData2(int data) { cout << this->data << endl; }
}
int main() {
      Thing t(4);
      t.printData1(5);
      t.printData2(5);
      return 0;
}
```

12. When is a destructor called? (two instances)
    a. When an object variable leaves scope
    b. When new memory is allocated on the head
    c. When delete is called with an object pointer
    d. When an object pointer leaves scope

13. True/False
    We can re-code any recursive function and turn it into a loop?

14. Why do we often use dummy head and tail nodes? (~1 sentence)

15. Write a loop to sum and return all values in a linked list that has dummy head and tail nodes. Do not add the head and tail values. The Node structure has an int data value and has next and prev Node pointers. This is a member function of the linked list class and we have a pointer to head and tail.

    ```
    int sumValues();
    ```

16. Write a recursive version of the same function at question 17. You'll need to use a recursive helper function.

CS162 Programming Final Instructions

Setup

1. ssh into syccuxas01.pcc.edu
2. copy tar into your directory, expand it, and go into directory
```
$ cp ~zachary.ankenman/final.tar .
$ tar -xf final.tar
$ cd final
```
3. Make sure it compiles and runs without modification
```
$ make run
```
4. Edit final.cc and add your name to the comment block at the beginning

Programming

`make run` will clean the old binary, create a new one, and run it!
Compare your answers to the expected output.

Fill out the body of these functions:

- ❏ `15 pts - double sumLoop(double* ptr, size_t size)`
- ❏ `15 pts - double sumRec(double* ptr, size_t size)`
- ❏ `5  pts - LinkedList()`
- ❏ `10 pts - ~LinkedList()`
- ❏ `10 pts - void printList()`
- ❏ `10 pts - void pushFront(double newData)`
- ❏ `15 pts - void pushBack(double newData)`
- ❏ `15 pts - double popFront()`
- ❏ `10 pts - int getLength()`
- ❏ `15 pts - void printHelper(Node *n)`
- ❏ `15 pts - int countNodes(Node *n)`
- ❏ `15 pts - Node* findEnd(Node *n)`

Turn in

1. Copy your midterm.cc to append your name
```
$ cp final.cc final_first_last.cc
```
2. Modify the permissions to make sure I can open it
```
$ chmod 666 midterm_first_last.cc
```
3. Copy it to my dropbox
```
$ mv final_first_last.cc ~zachary.ankenman/dropbox
```

Feel free to ask me to check the dropbox to ensure it got handed in properly.