CSCI 135 - Test 3

Name: _____

Section: _____

Instructions

- READ THESE DIRECTIONS!
- Clear your desk before the test (including any cellphones, electronic devices, etc.)
- Do not open this exam before you are told to do so.
- This is a closed-book closed-notes test.
- You may not use constructs or library functions not covered in class or specific to some C++ version. You do not need to mention header files, etc.
- You will not be graded on minor syntax errors as long as they don't make your answer ambiguous. Conversely, conceptual errors result in major deductions, even if syntactically minor.
- Budget your time well. The questions are not necessarily in order of difficulty or time!

1. (8%) Consider the following definition:

```
class SomeClass {
public:
  int      n1;
  int      *n2;
private:
  static   int n3;
  double   r1;
};
int main() {
  SomeClass x1,x2;
  // POINT 1
  SomeClass *x3;
  // POINT 2
  // CODE TO PART C GOES HERE
}
```

   Assume ints require 4 bytes, pointers require 6 bytes and doubles require 8 bytes.
   NO PARTIAL CREDIT WITHOUT EXPLANATION!

   (a) How much memory is allocated at POINT 1?

   (b) How much memory is allocated at POINT 2?

   (c) Write code to set x3's n2 member to 5 (at position stated above). Be careful!

2. (3%) Briefly explain what distinguishes a struct from an ADT (<2 sentences).

3. (24%) Write the following functions. You MUST use recursion for credit. You may use the string library functions: indexing, `length()`, `+`, and `substr(int n)` (recall that substr(n) returns a string with all but the first n characters).

   (a) `string alternate(string s)`: returns a string that contains every other element of s starting at the beginning. For example, `alternate(''dog'') = ''dg''`

   (b) `string uc(string s)`: returns a string that is the same as s except that every upper case character is replaced with a "UC".
   Hint: assume you have the uc of the string's tail.

   (c) `string uc2(string s, string repl)`: same as above, but replace it with repl instead of "UC".

4. (35%) Consider the following definitions:

```
class Book {
public:
  string getTitle();
  int    getPages();
  void   setTitle(string tt);
  void   setPages(int count);
  friend istream & operator
    >>(istream & istr, Book book);
private:
  NameList *authors;
  string   title;
  int      pages;
}
```

```
class NameList {
public:
  // following creates object from argument
  NameList(vector<string> names);
  ~NameList();
  ...
private:
  ...
};
```

Write the following assuming the above has already been written. You may NOT add/modify either class (other than what the problems ask for).

(a) Draw a solid box around each accessor function in `Book`. Also draw a dashed box around each mutator function in `Book`.

(b) Write a Book constructor that takes as arguments a title, page count, and a single author (in that order).

(c) Write a Book destructor.

(d) Write a `main` function that declares a collection of exactly 64 books named `library` and populates it from the input file 'data.txt" (you may assume the file has no errors). It should then print the title of the largest book. Finally, replace the largest book with one by author "doe", title "I live to code", and page count 256. You may assume that you correctly wrote all code from earlier parts.

5. (30%) Consider the following definition for a class that represents information about a container of something.

```
class SomeType {...};
class SomeClass {
public:
  int      size();           // number of elements in container
  SomeType largest();        // returns largest element
  void     removeLargest();  // removes largest element
  void     add(SomeType n);  // add object n to container
  SomeClass();               // create empty container
private:
  ...
};
```

You may assume that SomeClass assignment is overloaded to produce a copy (so can assign obj2=obj1 where both are objects in SomeClass). You may also assume that SomeType has overloaded versions of all standard arithmetic/comparison operators.

Write non-member functions with the following prototypes. You should NOT modify any object passed as argument.

(a) `SomeType secondLargest(SomeClass x)`: return second largest element of container.

(b) `SomeClass bigger(SomeClass x, SomeType n)`: return a container with all elements of x that are strictly greater than n.