University of Washington
Outreach Program

Name_____

1. Let's explore working with file input and output in our programs.
   a. Write a C++ program that opens a file (an arbitrary text file that contains alphas, digits, and punctuation) for input to your program. You can create this file using notepad. Read each character and write it to your display.
   b. Write a C++ program that opens a file (an arbitrary text file) for output from your program. Verify that you can write alphas, digits, and punctuation to the file.
   c. Write a C++ program that opens two files, for input and a second for output. Process each character that you read according to the following rules:
      1. If the character is an alpha, change its case...upper to lower and lower to upper
      2. If the character is a digit, subtract it from 9.
      3. All other characters remain unchanged.

   After processing, write each character to the output file.

2. Define a simple class with a default constructor, a constructor that takes a single parameter, and destructor. Annotate each one so that it writes to stdout each time it is invoked.

   Declare 4 functions, f1, f2, f3, and f4 with the following prototypes,

   ```
   void f1 (SimpleClass aSimpleClass);
   void f2 (SimpleClass aSimpleClass);
   void f3 (SimpleClass& aSimpleClass);
   void f4 (SimpleClass* aSimpleClassPtr);
   ```

   Declare an instance of SimpleClass in the function main and call f1 with that instance as an argument.

   Declare an instance of SimpleClass in the function f1. Pass that instance by value, to function, f2.

   Declare an instance of the SimpleClass in f2 that uses the parameterized constructor. Pass that instance by reference to function f3.

   Declare a function f4 that takes a pointer to an instance of SimpleClass. Call f4 from within f3 using a pointer to the argument passed into f3.

How many instances of your class were constructed? How many instances of your class were destructed? Why?

3. In the pervious homework, we began designing a new and improved (new and improved, huh…which is it?) type of array. The new array was specified to expand as data is entered, contract when it is removed and it was to support the following access methods.

   void put (int aValue) - puts aValue at array index 0.

   void atPut (int anIndex, int aValue) - puts aValue at array index anIndex – the array must expand when the index is beyond the size of the array. We have overwrite semantics when the index is in bounds and insert semantics otherwise.

   void atGet (int anIndex, int &aValue) - returns a copy the data at array index anIndex.

   void atRemove (int anIndex) - removes the data at array index anIndex – the array must contract.

   void showArray - displays the contents of the array.

   As the first steps towards the detailed design of the new type of array, we developed the use cases that specified the public interface to the data type and the class diagram identifying all of the public and private members.

   We now take the next and final steps to complete the design. To this end, implement and test your design for the array.