Lab 2

Course: CSE 165
Section: 04L
Due: Thursday, September 23, at 11:59 pm

All the exercises below are selected from the textbook: Thinking in C++ (volume 1).

1. [Exercise-15 on Page 228] Create a struct that holds two string objects and one int. Use a typedef for the struct name. Create an instance of the struct, initialize all three values in your instance, and print them out. Take the address of your instance and assign it to a pointer to your struct type. Change the three values in your instance and print them out, all using the pointer. [10 points]

2. [Exercise-26 on Page 229] Define an array of int. Take the starting address of that array and use static_cast to convert it into a void*. Write a function that takes a void*, a number (indicating a number of bytes), and a value (indicating the value to which each byte should be set) as arguments. The function should set each byte in the specified range to the specified value. Try out the function on your array of int. [15 points]

3. [Exercise-29 on Page 230] Modify FloatingAsBinary.cpp so that it prints out each part of the double as a separate group of bits. You'll have to replace the calls to printBinary( ) with your own specialized code. [25 points]

4. [Exercise-18 on Page 274] Write a function that takes a char* argument. Using new, dynamically allocate an array of char that is the size of the char array that's passed to the function. Using array indexing, copy the characters from the argument to the dynamically allocated array (don't forget the null terminator) and return the pointer to the copy. In your main( ), test the function by passing a static quoted character array, then take the result of that and pass it back into the function. Print both strings and both pointers so you can see they are different storage. Using delete, clean up all the dynamic storage. [20 points]

5. [Exercise-24 on Page 275] Create a struct that holds an int and a pointer to another instance of the same struct. Write a function that takes the address of one of these structs and an int indicating the length of the list you want created. This function will make a whole chain of these structs (a linked list), starting from the argument (the head of the list), with each one pointing to the next. Make the new structs using new, and put the count (which object number this is) in the int. In the last struct in the list, put a zero value in the pointer to indicate that it's the end. Write a second function that takes the head of your list and moves through to the end, printing out both the pointer value and the int value for each one. [30 points]

Requirements:

* Usage of spaces, blank lines, indention, and comments for readability

* Descriptive names of variables, functions, structs, classes, and objects (if any)

* Appropriate usage of structs, classes, and objects (if any)

Late Penalties:

* 10-point deduction per day late until zero