CS 120 - Lab #13

Due Friday, April 25th.

For this lab we will be creating several different functions that each swap the value of two variables. The first function will take the variables by value, the second will take the variables by reference, and the third will take pointers to the variables (where take means accept as arguments). The return types of these functions is up to you. As an example, if our variables were floats, and our unknown return type denoted as "?", the prototypes would look like this:

- `? swap_by_value(float, float);`
- `? swap_by_reference(float&, float&)`
- `? swap_by_pointer(float*, float*)`

For ease of readability and implementation, use a temporary value inside your function to facilitate the swap, rather than "fancy" arithmetic.

Example

Your main function will likely look like this:

```
#include <iostream>

using namespace std;

int main()
{
  float first = 13;
  float second = 42;
  float* first_ptr = &first;
  float* second_ptr = &second;

  cout << "Original pair: " << first << ", " << second << endl;
  swap_by_value(first, second); // may need to capture and use a returned object
  cout << "Reversed pair: " << first << ", " << second << endl;
  swap_by_reference(first, second); // should probably not need to return anything
```

```
  cout << "Re-reversed pair: " << first << ", " << second << endl;
  swap_by_pointer(first_ptr, second_ptr);
  cout << "Re-re-reversed pair: " << first << ", " << second << endl;
}
```

When swapping by value, note that you can only return a single object in C++, so cannot simply return two (swapped) variables. Also realize that since the function receives the variables by value, they are copied, and so changing them inside the function will not change the value of the variables originally sent. A potential solution may include the use of a simple struct object. It is okay if you do not get this to work, as swapping "by value" is an odd thing to do. Please note your attempt, and why think it did not work.

Swapping by reference is probably going to be the easiest function of this bunch, but you should see how it works.

Swapping by pointer is the focus of this lab, so prioritize it. Note that you do not necessarily have to dynamically allocate anything with `new`, and if not, will not need matching calls to `delete`.

Submission

Please turn in:

1. Your source code
2. Your tests (using script to record)
3. Explanations of functions that didn't work
4. Explanations of functions that did work