CS 121 SI – Week 2 Worksheet – Includes, Namespacing, and Functions

1. What is a preprocessor directive? Give an example of a preprocessor directive and what your example does.

   A preprocessor directive is a statement in the code that is ran before compile time (i.e. for the preprocessor to run). It is not actual C++ syntax.

   Examples: *#include <iomanip>* which "pastes" the iomanip library into your code for usage.
   #define PI 3.141592 which (unsafely) replaces all instances of PI with 3.141592

2. Assume you work for a company as a software migration expert and your project leader wishes to use the following program in the company's software suite:

```
#include <iostream>
#include <string>
#include <loginsys>  // made-up library used for calling validUser in this exercise
~~using namespace std;~~ // must omit or else conflicts with loginsys and iostream occur
using std::getline;      // implied that loginsys never uses this so this is valid
using std::string;       // “                                                  “
using std::endl;         // “                                                  “
int main()
{
        string user_acct = “”;
        int pin = -1;

        std::cout << “Please enter in your account user name: “;

        getline(std::cin, user_acct);

        std::cout << “Please enter in your account pin: “;

        std::cin >> pin;

        if ( validUser(user_acct, pin) )        // assume loginsys has this function defined
        {
                std::cout << “Thank you for logging into the system.” << endl;

                //... other code for stuff after logging in...
        }
        else
        {
                std::cout << “ERROR: Invalid user data submitted. Ending program...” << endl;
                return -1;          // arbitrary return code for this program (bad login info)
        }

        return 0;                   // returning 0 (historically) means no errors in the code
}
```

   Unfortunately for you, the library *loginsys* reserves the keywords *cout* and *cin* for its own special purposes. To keep your job, **tweak the code so your company may seamlessly use it.**

3. Write a program that does the following:
   - Takes three integer side lengths (a,b,c) from the user.
   - Determine the type of triangle the user supplied (right, equilateral, or neither)
   - Output the type of triangle the user supplied.
   - Request again if they would like to check another.

   An example output would look something similar to the following:

```
        Please enter three side lengths (separated by spaces): 3 4 5
        The side lengths provided form a right triangle.
        Another triangle (y/n)?: n
```

   To save space on this paper, the program does not need to do input validation (e.g. user entered char instead of int for a side length).

< There are three .cpp solutions for this problem, located in the solutions folder for this week.
The three types are:
- No functions used
- Functions used
- Improved & with functions >

4. In computer science, a function is known as a task or action. They sometimes take an input and return an output, but do not always have to do so. The format for declaring a function:

```
return_type function_identifier ( parameter_list_separated_by_commas )
{
        // statements in functions
        return return_value;
}
```

An example program:

```
#include <iostream>

double square ( double x )  // behaves identically to the math function f(x) = x^2
{
        double result = x*x;
        return result;
}

void printHello()          // a function that returns nothing (void) and no inputs (empty param list)
{
        std::cout << “Hello!\n”;
}

int main() // this is also a function
{
        std::cout << “4 squared is: “ << square(4.0) << std::endl; // outputs “4 squared is 16.0”
        printHello();                                              // outputs “Hello!”
        return 0;
}
```

Now, **convert the following math functions into C++ function declarations**. You will need to call a function from *cmath* to (easily) do the last one. Assume all variables are double-type parameters, and that the function returns a double as well.

$f(x)=mx+b$

```
// f(x) = m*x + b
double line (double m, double x, double b)
{
        return ( (m*x) + b )
}
```

$f(x)=(y-y_0)/(x-x_0)$

```
// f(x) = (y-y0)/(x-x0)
double slope (double y, double yknot, double x, double xknot)
{
        return (y – yknot)/(x - xknot);
}
```

<Rest of solutions on next page>

$$f(x)=x_0+V_0t+\frac{1}{2}a_xt^2$$

```
// f(x) = x0 + v0*t + .5 * ax * t^2
double position (double xknot, double vknot, double time, double ax)
{
        //for ease of readability
        double left_half = xknot + (vknot * time);
        double right_half = .5 * ax * time * time;

        return left_half + right_half;
}
```

$$f(x)=\sqrt{V_0^2+2a_x(x-x_0)}$$

```
// f(x) = sqrt(v0^2 + 2*ax * (x - x0) )
double position2 (double vknot, ax, double x, double xknot)
{
        // for ease of readability again
        double left_half = vknot * vknot;
        double right_half = 2.0 * ax * (x - xknot);

        return sqrt(left_half + right_half);
}
```

5. Looking back at problem 3: If you did not use functions where do you think they could exist? If you did use functions, list what functions you implemented.

Possible functions (both in and out of the program's scope):

- Verifying the type of triangle
- Handling user input (especially if user validation is a necessity)
- Displaying a menu (to improve the program's looks to the user)

6. Compare and contrast problems 3 & 5. Is using functions in problem 3 more convenient, less convenient, or makes no difference? Explain your thoughts.

For what was requested, a functional approach doesn't benefit too much. However, if I wanted to improve the program more (e.g. display a fancy menu, validate user input, add more triangle options, etc.) then functions would definitely make things more convenient (and easily manageable).