CS&E 1222

Lab 8 – Functions Lab Assignment – 20 points

- ✓ The *lab* must be accomplished solely by you:
  - DO NOT look at anyone's code other than your own, including code from another's student in your section or another section of the course, or any third party source, e.g. the Internet
  - DO NOT share or copy anyone else's code for any graded assignment
  - DO NOT work in pairs or groups
- ✓ All cases of academic misconduct will be reported to the *Committee On Academic Misconduct* (COAM).

Setting up the Programming Environment

Effective commenting and tabbing will affect your grade. The "style" of your program should follow the style of the sample programs in the lecture notes. Also see the example code from Lab #1. Your program should have the file name, your name, creation and last modification dates and a brief description of the program in the comments. ***In addition, read the document on "Commenting" found in the Content tab on Carmen under "Tutorials"***.

1. At the Linux command line type `mkdir lab8`. This will create a new directory named **`lab8`**. Work out of this directory. In order to do that, type `cd lab8`. This changes the current working directory to the directory **`lab8`**.
2. If you have created the directory **`lab8`**, then just type `cd lab8`.
3. Copy the files **`polarcoord_solution.exe`** and **`isprime_solution.exe`** from the directory **`/class/cse1222/9643/lab8`** into the current directory. Copy the files **`polarcoord_template.cpp`** and **`isprime_template.cpp`** using the following commands:

```
cp /class/cse1222/9643/lab8/polarcoord_template.cpp polarcoord.cpp
cp /class/cse1222/9643/lab8/isprime_template.cpp isprime.cpp
```

Be sure to include ***9643*** (this is your course section indicator) and the period, ".".

Programming Assignment

Add functions and function prototypes to the file **`polarcoord.cpp`** as specified below. When completed, the program reads in the polar coordinates of a point and computes and outputs the Cartesian coordinates. The main() routine of the program is already provided in **`polarcoord_template.cpp`** (which you copied into **`polarcoord.cpp`**.) DO NOT MODIFY ANY OF THE CODE in procedure main(). Your task is to add two functions, degrees2radians() and compute_coord(), so that the program produces the desired results.

Add functions and function prototypes to the file **`isprime.cpp`** as specified below. When completed, the program reads in a minimum and maximum integer and returns all prime numbers between the minimum and maximum. The main() routine of the program is already provided in **`isprime_template.cpp`** (which you copied into **`isprime.cpp`**.) DO NOT MODIFY ANY OF THE CODE in procedure main(). Your task is to add two functions, read_range() and is_prime(), so that the program produces the desired results.

Run **`polarcoord_solution.exe`** and **`isprime_solution.exe`** to see an example of the programs.

1. DO NOT MODIFY ANY OF THE CODE in procedure main().
2. All functions should be written AFTER the main procedure.
3. A function prototype should be written for each function and placed BEFORE the main procedure.
4. Each function should have a comment explaining what it does.
5. Each function parameter should have a comment explaining the parameter.
6. Note that **`polarcoord.cpp`** and **`isprime.cpp`** will NOT compile since the function prototypes and function definitions are missing.
7. Program **`polarcoord.cpp`**:
    a. Write a function degrees2radians() which converts degrees to radians. The function has one parameter: an angle in degrees of type double. The function returns a value of type double which is the angle in radians.
    The formula to convert degrees to radians is:

$$R = D \times \pi / 180$$

    where D is degrees and R is radians.

    b. Write a function compute_coord() which computes the Cartesian (*x*, *y*) coordinates of a point from its polar coordinates (*radius*, *angle*) where *angle* is measured in radians. The function has four parameters listed in the following order:
        i. the polar radius of the point,
        ii. the polar angle in radians of the point,
        iii. the x-coordinate of the point,
        iv. the y-coordinate of the point.

    All parameters should have type double. The first two parameters should be passed by value and the last two parameters should be passed by reference. The function does not return any value, but it modifies the last two parameters.

    The formula to compute the *x* and *y*-coordinates from the polar coordinates is:

$$x = radius \times \cos(angle),$$
$$y = radius \times \sin(angle),$$

    where *radius* is the polar radius, *angle* is the angle measured in radians, *x* is the *x*-coordinate and *y* is the *y*-coordinate of the point.

8. Program **`isprime.cpp`**:
    a. Write a function read_range() which reads in the minimum and maximum values. The function has two parameters listed in the following order:
        i. the minimum value of the range;
        ii. the maximum value of the range.

       Both parameters should have integer type. Both parameters should be passed by reference. The function does not return any value, but it modifies the two parameters.

       The function prompts for the minimum and maximum values. If the minimum or maximum values are less than 2, then the program prints an error message and prompts again for the two values. If the minimum value is greater than the maximum value, then the program prints an error message and prompts again for the two values.

       The program prints error messages and prompts for the minimum and maximum values until it gets two values greater than or equal to 2 and such that the minimum is less than or equal to the maximum.

    b. Write a function is_prime() which determines if a number is prime. The function has one integer parameter. The function returns **true** if the input parameter is prime and **false** otherwise.

       A number a is prime if and only if (a mod $b \neq 0$) for $b = 2, 3, ..., (a - 1)$.

9. Be sure to add the header comments "File", "Created by", "Creation Date" and "Synopsis" at the top of the file. Each synopsis should contain a brief description of what the program does.

10. Be sure that there is a comment documenting each variable.

11. Check that you did not delete or modify any of the original code in procedure main().

12. Check your output against the output from the solution executables provided.

13. Be sure that your *if* statements, *while* loops, and blocks are properly indented.

14. Be sure to add the header comments "File", "Created by", "Creation Date" and "Synopsis" at the top of the file. Each synopsis should contain a brief description of what the program does.

Submit Your Work

Important: Any program which does not compile and run will receive no credit!
If you are not sure what this means please ask your instructor.

Submit the file **`polarcoord.cpp`** and **`isprime.cpp`** using the *Lab 8* drop box on Carmen. **DO NOT** submit the file **`a.out`**. **DO NOT** submit uncompleted work from *In-Lab8*. This will not be graded.