CS1580 Lab 8

1 Lab Objectives

The objective of this lab is to:

1. Learn how to use **function templates**
2. Learn how to split code in **multiple files**

2 Task

Your program should take the radius of a circle from the user where it can be **int** or **float** or **double**. You can take the radius as input inside main function. You have to use three files:

1. lab8.cpp --------- **(contains the main() function)**
2. helper.cpp --------- **(contains body of the functions)**
3. helper.h ()--------- **(contains prototypes of the functions)**

Note that, it is encouraged to write the template functions on the .h file.

Use the following functions in the program:

- ***void greet();*** **// Output a Welcome statement.**
- ***T computeArea(T rad); // Computes the area of the circle***
- ***void printArea(T area); // print the values***
- ***void signout(); // a goodbye msg***

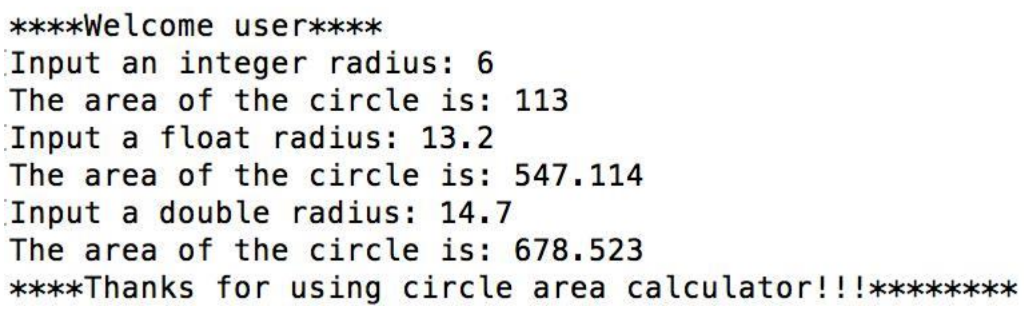
Write description, pre and post-conditions for each function as a part of the documentation. **Do not use function overloading. You must use templated functions. You should take at least 3 radii of different data types. See the sample output (I am considering value of pi as 3.14).**

3 Steps

1. Remotely connect to a Unix/Linux machine using Putty
2. Make a new directory named **lab8** under **cs1580** folder and go into the directory **lab8**.
```
cd SDRIVE/cs1580
mkdir lab8
cd lab8
```
3. using jpico, create lab8.cpp, helper.cpp and helper.h (jpico helper.h, helper.cpp). You can open multiple windows in putty and edit different files in different windows.
4. Declare the function prototypes for greet() and signout() in “helper.h” file
5. Define template functions computeArea() and printArea() in “helper.h” file
6. Define greet() and singout() in “helper.cpp” file

7. Include other necessary headers in each file
8. In helper.cpp and lab8.cpp, include “helper.h” file at the top as followed:
   ```
   #inlcude “helper.h”
   ```
9. Compile and run the program
   ```
   fg++ *.cpp -o run
   ```
10. Submit your program
    ```
    cssubmit 1580 <section> <assignment number>
    ```
11. Exit

4 Sample output:



```
****Welcome user****
Input an integer radius: 6
The area of the circle is: 113
Input a float radius: 13.2
The area of the circle is: 547.114
Input a double radius: 14.7
The area of the circle is: 678.523
****Thanks for using circle area calculator!!!********
```

5 Constraints

1. Do not use function overloading. You must use the templated function.
2. Declare the function prototypes and define the template functions in “helper.h” file
3. Define other functions (greet() and signout()) in “helper.cpp” file
4. You should take at least 3 radius as input for different data types (i.e. int, float, double) and show as output the computed area
5. Use function documentation for all the functions in the “helper.h” file
6. **Use return statement even in void functions**

6 Hints

1. **Documentation:**

   Documentation rules:
   A function description: Concise statement about what the function does (not how)
   - A precondition: What must be true of the arguments/parameters in order to guarantee the post condition
   - A post condition: States what is true after as a result of success.

- Example

```
// Description: cylVol() calculates and returns the
//              volume of the cylinder
// Pre: The rad and ht must be positive
// Post: Returns the volume of the cylinder

double cylVol(const double rad, const double ht);
```

2. **Template function:** The following template function swaps two parameters passed to it

```
template <typename T>
void mySwap(T &val1, T &val2)
{
  T temp = val1;
  val1 = val2;
  val2 = temp;

  return 0;
}
```

7 How to get full points

1. Fill out all the information in the program header, i.e.

```
// Programmer:          <Your Name here>
// MST Username:        <Your MST Username here>
// Instructor Name:     <Your Instructor's here>
// Section:             <Your Section>
// Date:                <Today's date>
// File:                <Name of your cpp file>
// Purpose:             <Description of what it does>
```

2. The contents of the main function are indented 2 spaces.
3. Curly braces { and } go on their own lines.
4. Use meaningful variable names.
5. Write comments where it is needed
6. Make sure your program follows all the constraints discussed in section 5
7. Program compiles and runs correctly
8. You have submitted your program correctly