CS 1136 Laboratory Lesson #4b

Functions

Important Information

This is your 4th lab lesson for the semester and will use the following guidelines.

Late submissions will get 0 points. If you save it as a draft and do not actually click on submit you will get 0 points. If you submit an empty file you will get 0 points. If you submit something other than the source files (for either the text files required or for the C++ source) you will get 0 points. Any other condition that results in incorrect or incomplete submissions will be graded as is and no changes after the due date will be allowed.

Anything you submit after the due date will be ignored.

Submissions must be made via eLearning. You can make multiple submissions and the final version you submit – before the deadline – will be used.

Exercise 1: A simple function

There one review file this week. The file **CS 1136 Laboratory Lesson 4b function review.pdf** gives a brief review of functions in C++.

Problem description

The following complete program contains a function that sums the numbers 1-10 inclusive.

We want the main to display the sum of the values computed in the compute_sum function. As it exists today this sum does not display correctly inside main.

```
// Lab 4b Exercise 1
// The compute_sum function
//
// Program by:     Place your name here
#include <iostream>

using namespace std;

// Compute the sum of all of the numbers from 1 to n where n
//  is a natural number
// use the formula:  n(n+1)/2
void compute_sum(int limit)  // compute_sum function
{
      int sum_to_limit;
```

```
      sum_to_limit = limit * (limit + 1) / 2;
}

int main()
{
   int sum = 0;
   int maxNumber;

   // get the maxNumber for the function call
   cout << "Enter a whole number greater than 0" << endl;
   cin >> maxNumber;

   // call compute sum
   compute_sum(maxNumber);    // Call to compute_sum function
   // display the sum calculated by the compute_sum function
   cout << "The sum of 1 to " << maxNumber;
   cout << " is " << sum << endl;

   return 0;
}
```

Update this program and call it **Lab4b_Exercise1.cpp.** Currently the `compute_sum` function calculates the sum but does not return the value back to the calling function. You must update `compute_sum` to return the sum to the calling function. The `compute_sum` function MUST NOT output the value to the display. Your `main` function needs to prompt for maxNumber and pass that value to compute_sum. In addition your main function needs to be updated so it can save the value returned from `compute_sum` and display the value. The `main` function must NOT CALCULATE the sum, this must be done by `compute_sum`. To summarize: the sum must be displayed by the main function, but it is calculated in the compute_sum function.

Do not use global variables as part of your solution. All values needed by the functions need to be passed as parameters. Any values returned by the functions need to be returned via the return statement.

The problem with the current, incorrect, version of the program is that the `compute_sum` function computes the sum, but does not return it back to the `main` function.

Here are the questions you need to answer to update this program:

- What value is computed in the `compute_sum` function that the `main` function needs?
- How can we return the value back to the `main` from the `compute_sum` function?
- How can up update the `compute_sum` function so it returns back the sum it has calculated?
- What changes need to be made to the `main` function to use this returned value?

Note: If you are making major changes to the main and compute_sum funtions you are probably doing way too much work. See your instructor, lab assistants or CS Tutors for help.

Once updated the program should output the current value of sum. Here is a sample run:

```
Enter a whole number greater than 0
10
The sum of 1 to 10 is 55
```

1. Create an empty project with the name **Lab 4b Exercise 1 Project.**

2. Create your program with the name **Lab4b_Exercise1.cpp**. This should originally have the code in the Problem description. This is also in file **Lab4b_Exercise1_template.cpp** on eLearning.

3. Make sure the following comments are at the beginning of your program. Replace **Place your name here** with your actual name.

```
// Lab 4b Exercise 1
// The compute_sum function
//
// Program by:     Place your name here
```

4. **Make sure you use meaningful names in your program. Also you should include comments to describe what your program is doing. You can do this by including the comments and variable names from the original code above.**

5. See the problem description above and the sample output for guidance on what is needed for the output of your application. The problem description also shows the output to be expected from the updated program.

6. Test and debug your program. Make sure you don't use global variables and that the sum is only calculated in the compute_sum function but is output by the main.

7. See the grading guidelines in **Part 3**. You will submit the C++ source file (.cpp file) to eLearning. You will submit the **Lab4b_Exercise1.cpp** file in **Part 4.**

Part 2: You cannot beat this exercise

Problem description

Your Target Heart Rate is the rate at which your heart should beat to get the maximum benefit from aerobic exercise. This rate is generally agreed to be 60 to 70 percent of your maximal heart rate. Your maximal heart rate equals 220 minus your age.

Create a program which includes the following:

The following calculations need to be done in your program:

Maximum heart rate = 220 – age

Minimum target heart rate = Maximum heart rate * 60%

Maximum target heart rate = Maximum heart rate * 70%

You want your heart beats per minute to be between the Minimum target heart rate and the Maximum target heart rate.

Create a program which includes the following four functions:

- The `calculateMaximumHeartRate` function. The body for this function has been written for you. You will have to fill in the body of the function with the actual processing.
- The `displayTargetHeartRate` function. You will need to write this function.
- The `processTargetHeartRate` function. You will need to write this function.
- The `main` function. This function has been written for you.

The template for this exercise can be found in file **Lab4b_Exercise2_template.cpp** on eLearning.

Here is the processing needed for the three functions you have to write.

- The `calculateMaximumHeartRate` function takes one input parameter, an `int` value for the age. The function returns an `int`, the calculated maximum heart rate.

  The pseudo code for the calculation is:

  Maximum heart rate = 220 – age

- The `displayTargetHeartRate` function takes one input parameter of type `int`. This is the maximum heart rate. The function does not return back a value.

  The function should calculate the minimum target heart rate and the maximum target heart rate. The two target heart rates should then be displayed.

Here is the output for a maximum heart rate of 100. The output should display one decimal digit for the minimum and maximum target heart rates (see the output below).

```
Minimum target heart rate is 60.0
Maximum target heart rate is 70.0
```

- The `processTargetHeartRate` function does much of the work driving the application.

  The function should ask the user to enter an age.

  Here is a sample of the prompt and input value (input value is in **bold**):

  ```
  Enter your age:
  20 [Enter]
  ```

  Next the function needs to pass the age to the `calculateMaximumHeartRate` function and save the result of the function in a variable.

  The resulting value from the `calculateMaximumHeartRate` function needs to be passed to the `displayTargetHeartRate` function so it can output the minimum and maximum target heart rates.

- The `main` function calls the `processTargetHeartRate` function three times and then returns back to the operating system. The `main` function has been written for you.

Name your program **Lab4b_Exercise2.cpp**.

Here is a sample output from a program that meets the requirements.

Example:

```
Enter your age:
21 [Enter]
Minimum target heart rate is 119.4
Maximum target heart rate is 139.3
Enter your age:
25 [Enter]
Minimum target heart rate is 117.0
Maximum target heart rate is 136.5
Enter your age:
33 [Enter]
Minimum target heart rate is 112.2
```

```
Maximum target heart rate is 130.9
```

1. You should create your Empty Project with the name **Lab 4b Exercise 2 Project.**

2. Create your source file and give it a file name of **Lab4b_Exercise2.cpp**.

3. Make sure the following comments are at the beginning of your program. Replace **Place your name here** with your actual name.

```
// Lab 4b Exercise 2
// Determine the minimum and maximum target heart rates.
//
// Program by:     Place your name here
```

4. **Make sure you use meaningful names in your program. Also you should include comments to describe what your program is doing. You can use the problem description as a starting point for this. Put comments throughout your code.**

5. See the problem description above and the sample output for guidance on what is needed for the output of your application. The problem description also suggests a number of sample input values you should use for your application testing. You will need multiple test runs of your application to make sure your program is running properly.

6. Your application output does not have to match the output shown in the problem description, but all of the information shown above must be displayed in some way.

7. Test and debug your program. Try it with different, valid, values for the tests. Some sample run values are shown in the description above.

8. You will submit the C++ source file (.cpp file) to eLearning. See the grading guidelines in **Part 3**. You will submit the **Lab4b_Exercise2.cpp** file in **Part 4**.

Part 3: Here is the general grading we will be using for lesson 4b.

Note that you MUST submit working code with correct logic or points will be deducted. If you are having problems see your lab assistants, the tutors in the open lab and your instructor for CS 1136.

Lab 4b Exercise 1 is worth 30 points.

To get any credit you must upload the Lab4b_Exercise1.cpp file (the actual C++ source file).

Any syntax errors in your program will result in a minimum 5 point automatic reduction. You need to make sure you submit working programs.

Significant logic errors will result in a minimum 10 point reduction. Minor logic errors will result in a minimum 5 point reduction.

Not using meaningful variable names will result in a 5 point reduction. Failure to include comments will result in a 5 point reduction. There must be more comments that the required ones at the top of the application.

Note that the lowest grade you can get for exercise 1 is 0.

Lab 4b Exercise 2 is worth 70 points.

To get any credit you must have the Lab4b_Exercise2.cpp file (the actual C++ source file).

Any syntax errors in your program will result in a minimum 10 point automatic reduction. You need to make sure you submit working programs.

Significant logic errors will result in a minimum 20 point reduction. Minor logic errors will result in a 5-15 point reduction. Minor typos will result in a 5-10 point reduction

Not using meaningful variable names will result in a 5 point reduction. Failure to include comments will result in a 5 point reduction. There must be more comments than the required ones at the top of the application.

Note that the lowest grade you can get for exercise 2 is 0.

Part 4: Submit your files to eLearning

1. Go to elearning.utdallas.edu and submit your files for lab assignment 4b. The following files need to be uploaded. You need to upload these in one attempt. You can submit multiple attempts, but each attempt needs to contain all of the files.

   Lab4b_Exercise1.cpp
   Lab4b_Exercise2.cpp

2. **Make sure you submit all of your source files (the two C++ source files). You need to make sure you do this BEFORE the deadline. Late submissions will NOT be accepted.**

3. Validate that you have uploaded the correct files, the correct version of the files, and that your submission is not in draft status. If you change one of the programs create a new

submission with both programs.

4. Fill out the CS 1136 Laboratory Lesson Verification.txt file and upload it as a new submission. Failure to do this will result in a 10 point reduction in your grade.

5. Not having all of your submitted source files in one submission will result in a 10 point reduction. The verification document can be in a different submission.