

User-defined functions

Fun with numbers

The first part requires you to modify your program for assignment 5 by incorporating your own functions. You will read two numbers from an input file and do some processing on the numbers that lie between these two numbers.

Sample input data:

Here is a sample input file, named `twoNumbers.txt`, for the program.

```
First Number: 2
Second Number: 32
```

There is a single space character ' ' between the : and the numbers on each line.

Output

There are five sets of values to be output to a file called `processedNumbers.txt`

1. The set of prime numbers that lie between the two numbers (Note: Only the number 2 is a valid prime number. An odd number is a valid prime number if it is not divisible by any odd integer less than or equal to the square root of the number),
2. The sum of even numbers,
3. The sum of odd numbers,
4. The sum of squares of the even numbers, and
5. The sum of squares of the odd numbers.

Sample output:

Here is a sample output file, named `processedNumbers.txt`, which corresponds to the input data given above:

```
Programmer: Mir Farooq Ali
CS 1044 Spring 2004 Project 6-a

The prime numbers between 2 and 32 are 3 5 7 11 13 17 19 23 29 31

Sum of Even numbers
238
Sum of Odd numbers
255
Sum of squares of Even numbers
4956
Sum of squares of Odd numbers
5455
```

The first two lines identify the programmer and project, as usual. Then there is a blank line, followed by the set of prime numbers. This is followed by a blank line, followed by the various sums as indicated. You do not have to use both the while and the for loops for this project. Feel free to use any one of those two to generate the output as indicated above. Note that there is only one set of numbers for this project.

If you have read the description of how the Curator scores your program in the *Student Guide*, you know that is important that you use the same spelling and capitalization for all the labels shown above. The horizontal spacing does not effect scoring unless you combine things that should be separate or separate things that should be combined. You are free to experiment with the horizontal layout but you should try to align the columns neatly.

Note that you must insert blank lines and divider lines as shown.

User-defined function details

You are supposed to implement five functions in your program. The five function definitions are given below:

1. `bool isPrime(int number);`
// This function should return true or false depending on whether the specified parameter is a prime number or not.
2. `int sumOdds(int first, int second);`
// This function should return the sum of all odd numbers lying between the specified numbers, excluding first and second themselves.
3. `int sumEvens(int first, int second);`
// This function should return the sum of all even numbers lying between the specified numbers, excluding first and second themselves.
4. `int sumOddSquares(int first, int second);`
// This function should return the sum of squares of all odd numbers lying between the specified numbers, excluding first and second themselves.
5. `int sumEvenSquares(int first, int second);`
// This function should return the sum of squares of all even numbers lying between the specified numbers, excluding first and second themselves.

Mathematical Calculations:

In order to determine if a number is even or odd, you can use the modulo (%) operator. If a number modulo 2 is equal to 0, then that number is an even number. Otherwise, it is an odd number. Also, you can use the math library function `sqrt` to calculate the square root of a number. You have to include the `<cmath>` header file to be able to use the `sqrt` function. Also note that the `sqrt` function expects a double as its argument. So you need to cast your integer variable as a double before using it. Sample usage of the `sqrt` function is given below.

```
#include <cmath>
...
int number = 49;
int squareRoot;
squareRoot = (int)sqrt((double)number);
...
```

Evaluation:

Everything that was said in the specification for the earlier about testing still applies here. Do not waste submissions to the Curator in testing your program! There is no point in submitting your program until you have verified that it produces correct results on the sample data files that are provided. If you waste all of your submissions because you have not tested your program adequately then you will receive a low score on this assignment. You will not be given extra submissions.

Your submitted program will be assigned a score, out of 100, based upon the runtime testing performed by the Curator System. We will also be evaluating your submission of this program for documentation style and a few good coding practices. This will result in a deduction (ideally zero) that will be applied to your score from the Curator to yield your final score for this project.

Read the *Programming Standards* page on the CS 1044 website for general guidelines. You should comment your code in the same manner as the code given for the first two programming assignments. In particular:

- You should have a header comment identifying yourself, and describing what the program does.
- Every constant and variable you declare should have a comment explaining its logical significance in the program.

- Every major block of code should have a comment describing its purpose.
- Adopt a consistent indentation style and stick to it.

Your implementation must also meet the following requirements:

- Choose descriptive identifiers when you declare a variable or constant. Avoid choosing identifiers that are entirely lower-case.
- Use named constants instead of literal constants when the constant has logical significance.
- Use C++ streams for input and output, not C-style constructs.
- Use C++ string variables to hold character data, not C-style character pointers or arrays.

Understand that the list of requirements here is not a complete repetition of the *Programming Standards* page on the course website. It is possible that requirements listed there will be applied, even if they are not listed here.

Submitting your program:

You will submit this assignment to the Curator System (read the *Student Guide*), and it will be graded automatically. Instructions for submitting, and a description of how the grading is done, are contained in the *Student Guide*.

You will be allowed up to five submissions for this assignment. Use them wisely. Test your program thoroughly before submitting it. Make sure that your program produces correct results for every sample input file posted on the course website. If you do not get a perfect score, analyze the problem carefully and test your fix with the input file returned as part of the Curator e-mail message, before submitting again. The highest score you achieve will be counted.

The *Student Guide* can be found at: <http://ei.cs.vt.edu/~eags/Curator.html>

The submission client can be found at: <http://eags.cs.vt.edu:8080/curator/>

Pledge:

Each of your program submissions must be pledged to conform to the Honor Code requirements for this course. Specifically, you **must** include the following pledge statement in the header comment for your program:

```
// On my honor:  
//  
// - I have not discussed the C++ language code in my program with  
//   anyone other than my instructor or the teaching assistants  
//   assigned to this course.  
//  
// - I have not used C++ language code obtained from another student,  
//   or any other unauthorized source, either modified or unmodified.  
//  
// - If any C++ language code or documentation used in my program  
//   was obtained from another source, such as a text book or course  
//   notes, that has been clearly noted with a proper citation in  
//   the comments of my program.  
//  
// - I have not designed this program in such a way as to defeat or  
//   interfere with the normal operation of the Curator System.  
//  
// <Student Name>
```

Failure to include this pledge in a submission is a violation of the Honor Code.