**0. Math repetitor, version 1**

Redo exercise 4 of lab #3 with the following changes:

* the user can still submit the answer of the proposed addition and the program will tell her/him if this answer is correct
* the program will keep display yet another addition until the user enter the value 'y' or 'Y' as the answer to the question 'more addtion?'

Here is a log of execution of this program (user input is in bold):

Welcome to the Math tutor  
  
  178  
+ 360  
----- Enter the answer below:  
  **538**  
  
Congratulations!  
The answer is 538  
  
more addition (enter 'y' or 'Y')? **y**  
  
  388  
+ 481  
----- Enter the answer below:  
  **769**  
  
Sorry that is incorrect.  
The answer is 869  
  
more addition (enter 'y' or 'Y')? **y**   
  260  
+ 354  
----- Enter the answer below:  
  **614**  
  
Congratulations!  
The answer is 614  
  
more addition (enter 'y' or 'Y')? **n**Thank you for using Math Tutor

You must start from your version of exercise 4 from lab #3 and try to adapt it. The more suitable loop construct for this problem is the do-while loop because when the program starts, it displays at least one addition for the user.

**1. Math repetitor, version 2**

Redo previous exercise with the following change : now the program will keep display yet another addition until the user enter the value 0 as the answer. So 0 is a sentinel value used to express that the user wants to stop the program.

Here is a log of execution of this program (user input is in bold):

Welcome to the Math tutor  
  
  178  
+ 360  
----- Enter the answer below (or 0 to leave the Math Tutor):  
  **538**  
  
Congratulations!  
The answer is 538  
  
  388  
+ 481  
----- Enter the answer below (or 0 to leave the Math Tutor):  
  **769**  
  
Sorry that is incorrect.  
The answer is 869  
  
  260  
+ 354  
----- Enter the answer below (or 0 to leave the Math Tutor):  
  **0**  
  
Thank you for using Math Tutor

For this program, you should use the form:

while (true)  
{  
  ....;   
  
  if ( ... )  
     break;  
  ....  
}

to make the writing of the code more convenient.

**2. Number average, version 1**

As an extended version of exercise 1 of lab #3, write a program with a loop that lets the user enter a series of integers. The user should enter the value 0 to signal the end of the series. After all the numbers have been entered, the program should display the largest and smallest numbers entered together with the total number of integer entered and finally the average of these values (a double). Here is an example of execution of this program(user input is in bold):

You will enter a serie of integers and I will then tell you  
the largest and the smallest numbers you entered  
Enter 0 to terminate the series  
  
Enter an integer (0 to quit): **7**  
Enter another integer (0 to quit): **2**  
Enter another integer (0 to quit): **8**  
Enter another integer (0 to quit): **-4**  
Enter another integer (0 to quit): **9**  
Enter another integer (0 to quit): **5**  
Enter another integer (0 to quit): **0**  
  
The greatest number you entered is 9  
The least number you entered is -4  
The average of the 6 number(s) you entered is 4.5

Notice that the value 0 is not considered as one of the entered value. The average should be displayed with one decimal position.  
  
If the user enter the value 0 as the very first value, the program displays a specific message because it is not possible to define the minimum and maximum if there is no value:

You will enter a serie of integers and  
I will then tell you the largest and  
the smallest numbers you entered  
Enter 0 to terminate the series  
  
Enter an integer (0 to quit): **0**  
  
You didn't enter any value

**3. Temperature conversion**

In exercise 3 of lab #2 you were asked to write a program that converts a Celsius temperature to Fahrenheit. Write another program so it uses a loop to display a table of a range of Celsius temperatures and their Fahrenheit equivalents. The range is given by the user as two integers. If the first value is less or equal than the second, the tempatures appear in the table in increasing order, else there are displayed in decreasing order. Here is an example of execution of this program (user input is in bold):

Temperature Conversion Table  
  
Enter the range of Celsius temperatures to convert: **5 15**  
  
Celsius      Fahrenheit  
5            41.0  
6            42.8   
7            44.6     
8            46.4       
9            48.2    
10           50.0      
11           51.8    
12           53.6   
13           55.4       
14           57.2    
15           59.0

If the two values given by the user are equal, the program should display a table with only one line.

**4. Guessing game**

Write a program that generates a random number in the range 1 to 100 inclusive and asks the user to guess what the number is. If the user’s guess is higher than the random number, the program should display the message Sorry, that’s too high. If the user’s guess is lower than the random number, the program should display the message No, that’s too low. The program should use a loop that repeats until the user correctly guesses the random number. The program must keep a count of the number of guesses that the user makes. When the user correctly guesses the random number, the program should display the number of guesses. Here is an example of execution of this program (user input is in bold):

I’m thinking of a number...  
  
Can you guess what it is? **50**  
Sorry, that’s too low.  
Guess again: **75**  
Sorry, that’s too high.  
Guess again: **62**  
Sorry, that’s too high.  
Guess again: **56**  
  
Congratulations! You guessed it!  
I was really thinking of the number 56.  
You got it right in 4 guesses.

To debug your code, you should fix the secret number to a constant value (not a random value). You must ensure that everytime your program is executed, a new random number (different from the previous program execution) has to be guessed. Optional fun: find a simple strategy to ensure that you can guess the random number in no more than 7 guesses...

**5. Prime numbers**

A prime number (or a prime) is a natural number greater than 1 that has no positive divisors other than 1 and itself. Write a program that prompts the user for an integer greater than 1 and tells if this integer is a prime number or not. The program should keep asking for a number until the user enters the value 0 to exit. To check if a number n is prime, you should use a straightforward algorithm which consists of testing whether n is a multiple of any integer between 2 and square root of n inclusive. In addition, your program should be smart enough to avoid unnecessary testing (think about even numbers). When the number entered is not a prime, your program should tell why (i.e. display a divisor of this number). Here is an example of execution of this program (user input is in bold):

Welcome to the Prime Checker  
  
Enter a positive integer (0 to exit): **17**  
17 is prime  
  
Enter a positive integer (0 to exit): **28**  
28 is not prime because is a multiple of 2  
  
Enter a positive integer (0 to exit): **55**  
55 is not prime because is a multiple of 5  
  
Enter a positive integer (0 to exit): **127**  
127 is prime  
  
Enter a positive integer (0 to exit): **0**  
  
Thank you for using the Prime Checker

You need to round up square root of n using the function ceil from the cmath library.

**6.  Number average, version 2**

Redo exercise 2 of this lab, but in this new version, the integer values are read from a file (and not as input from the user). In this version, the value 0 is no longer used to signal the end of the the series of number. The program stops reading numbers when it reaches the end of the file and then displays the minimum, maximum, number of intergers read and the average of these intergers. You can check your program using the two following files:  [numbers-1.txt](http://users.polytech.unice.fr/~gaetano/db/cse142/labs/lab04/numbers-1.txt)  and  [numbers-2.txt](http://users.polytech.unice.fr/~gaetano/db/cse142/labs/lab04/numbers-2.txt)  
Here is a first example of execution of this program (user input is in bold):

Enter the filename: **numbers-1.txt**  
  
The maximum is 10  
The minimum is 1  
The average is 5.50

Here is a second example of execution of this program (user input is in bold):

Enter the filename: **numbers-2.txt**  
  
The maximum is 100  
The minimum is -40  
The average is 45.83

Unlike in the examples, you probably have to provide the full file name for the number files.

**7. Rainfall**

Write a program that reads some monthly rainfall data from a file and prints the data read followed by the average of all the rainfall. Each line in the file has a string (the name of the month) followed by a space character, followed by a double value (the rainfall for that month). The number of lines in the file is not fixed. You can check your program with the following file: [rainfall.txt](http://users.polytech.unice.fr/~gaetano/db/cse142/labs/lab04/rainfall.txt)  
Here is an example of execution of this program (user input is in bold):

Enter the filename: **rainfall.txt**  
  
     March   20.50  
     April   24.80  
       May   18.40  
      June   12.90  
      july    7.20  
    August   13.60  
 September   15.10  
   October   19.50  
  
The average rainfall is 16.50 mm

Unlike in the example, you probably have to provide the full file name for the rainfall file.