**1. Number average**

Redo exercise 1 of lab #2 using now an array to store the values input by the user. Your program should work as follow:

* the number of integer to be read and analyzed is fixed and defined in a constant (for example, 5)
* in a first part, your program read the (five) numbers and store them inside an array of the suitable size (five).
* in a second part, your program traverses the array and compute the least, the greatest and the average of all numbers
* in a last part, your program displays the results

In addition, your program should run correctly in case we change the value of the constant defining the number of integers input by the user. You should start from exercise 1 of lab #2 and try to adapt it.

**2. Date**

Write a program which reads a date from the user in the format year/month/day and display this date with the month in English. Here is an example of execution of the program (user input in bold):

Welcome to the date maker  
  
Enter the date in the form year/month/day or 0 to quit: **2017/4/15**  
The date is: April 15, 2017  
  
Enter the date in the form year/month/day or 0 to quit: **2017/12/31**  
The date is: December 31, 2017  
  
Enter the date in the form year/month/day or 0 to quit: 0  
Thank you for using the date maker

In addition, your program should detect wrong input values:

Enter the date in the form year/month/day or 0 to quit: **-2015/4/15**  
the year is wrong  
  
Enter the date in the form year/month/day or 0 to quit: **2015/14/2**  
the month is wrong  
  
Enter the date in the form year/month/day or 0 to quit: **2015/2/30**  
the day is wrong

On the previous log of execution, you can see that the program is detecting a wrong year value (negative number), a wrong month value (14) and finally a wrong day value (30 although February only have 28 days). We assume in this exercise there is no leap year (i.e. February has always 28 days). You should handle two arrays, one for the month names and one for the number of days in each month. In addition you may need to use the function ignore: check yourself the internet to see how to use it!

**3. Temperatures**

Write a program which reads from the user the average temperatures for each month of the year, and displays the average of those temperatures plus the list of months when the temperature was above average. Here is an example of execution of the program (user input in bold):

Welcome to the temperature checker  
Enter temperature for January: **8.6**  
Enter temperature for February: **10.7**  
Enter temperature for March: **12.4**  
Enter temperature for April: **14.7**  
Enter temperature for May: **16.3**  
Enter temperature for June: **20.5**  
Enter temperature for July: **25.7**  
Enter temperature for August: **28.4**  
Enter temperature for September: **23.6**  
Enter temperature for October: **18.2**  
Enter temperature for November: **14.8**  
Enter temperature for December: **13.2**  
  
The average temperature is 17.3  
  
Temperature in June was above average (20.5)  
Temperature in July was above average (25.7)  
Temperature in August was above average (28.4)  
Temperature in September was above average (23.6)  
Temperature in October was above average (18.2)  
  
Thank you for using the temperature checker

All the double values should be displayed with one digit in the decimal part. You should use two arrays, one for the month names (say, monthName) and one for the temperatures (say, temperature). Those two arrays are said to be parallel arrays because for each valid index i, temperature[i] is the average temperature of month i which name is monthName[i], i being in the range [0,11].

**4. Prime numbers**

Write yet another program to compute and display the n first prime numbers. Your program should implement  
the Sieve of Erathostene algorithm described as follow:

*1. initialize a list of primes with number 2*  
*2. initialize first candidate to 3*  
*3. traverse the list to look for a divisor of the candidate:*

*• if there is no divisor in the list, then the candidate is the next prime and you can insert at it at the*  
*end of the list*  
*• if there is a divisor in the list, the candidate is not prime*

*4. if the list contains n primes, the program ends after printing out the list of the n primes, else, pick up the*  
*next candidate and go to step 3*

The value n will be defined as a constant in the program (say, 100). You will use an array (of size n) to implement the list. Additionnaly, you will need to manage an integer variable count to hold the actual number of primes in the list (i.e. the length of the prefix in the array which holds the prime numbers). Your program should have the following functions in addition to the main:

* a function to check if a candidate is a multiple of some prime in the list
* a function to display the prime numbers at the end of the program

Remember that you don’t need to check for divisors by checking all the primes in the list (why?).

**5. Automatic grading**

You are to write a program to automatically grade exams. An exam consists of exactly 20 multiple-choice questions. Each question has one of four possible answers: A, B, C, or D. The first line contains the answer to the first question, the second line contains the answer to the second question, and so forth. Your program must read the contents of a file containing the correct answers (provided by the professor) into a char array, and then read the contents of another file (provided by the student) containing student’s answers, into a second char array. The program should then determine the number of questions that the student missed, and then display the following:

* A list of the questions missed by the student, showing the correct answer and the incorrect answer provided by the student for each missed question
* The total number of questions missed
* The percentage of questions answered correctly
* If the percentage of correctly answered questions is 70% or greater, the program should indicate that thestudent passed the exam. Otherwise, it should indicate that the student failed the exam.

Use the provided files to checks your code.  
  
**Supporting files**

* [grading-sheet.txt](http://users.polytech.unice.fr/~gaetano/db/cse142/labs/lab06/grading-sheet.txt)
* [correct-answers.txt](http://users.polytech.unice.fr/~gaetano/db/cse142/labs/lab06/correct-answers.txt)
* [student1-answers.txt](http://users.polytech.unice.fr/~gaetano/db/cse142/labs/lab06/student1-answers.txt)
* [student2-answers.txt](http://users.polytech.unice.fr/~gaetano/db/cse142/labs/lab06/student2-answers.txt)
* [lab06-ex5-log1.txt](http://users.polytech.unice.fr/~gaetano/db/cse142/labs/lab06/lab06-ex5-log1.txt)
* [lab06-ex5-log2.txt](http://users.polytech.unice.fr/~gaetano/db/cse142/labs/lab06/lab06-ex5-log2.txt)

**6. Displaying grades**

Write a program to display grades of students in a course. The grades are collected in a file with the following structure:

* the first line of the file holds the name of the course, like *C++ Programming*
* the rest of the file consists of S blocks, one per student. Each block is made of:
  + one line with the name of the student, like *Spiderman*
  + M lines with one double value each representing the M grades obtained by the student

Your program should read such a file and then prompt the user for the name of a student. If the user input the name of a student, then the program displays the grades of this student. If the user inputs the keyword stop then the program terminates. Here is an example of execution of this program (user input in bold):

Welcome to the grade database  
  
Enter the grading file name: grade-sheet.txt  
Enter the name of the student or stop to exit: **Hulk**  
Results for Hulk: 4.9, 6.7, 7.2, 5.3, average = 6.03  
  
Enter the name of the student or stop to exit: **Marc**  
Sorry, student Marc is not in our database  
  
Enter the name of the student or stop to exit: **Spiderman**  
Results for Spiderman: 7.50, 8.60, 9.20, 6.70, average = 8.00  
  
Enter the name of the student or stop to exit: stop  
Thank you for using the grade database

The values of S and M must be given as constants in the program. You must use S = 5 and M = 4 to match the format of the file grading-sheet.txt provided as a supporting file) which means that the corresponding course (”C++ Programming”) had 5 students and each student got 4 marks.  
  
To solve this problem, you should read the file and then store the information you read into two different arrays, one array for the names of the students (say, names) and one array containing arrays of double values (say, marks): this one is a two-dimensional array to store the marks for each student. Those two arrays are againparallel arrays just like in exercise 3 because for each valid index i, marks[i] is the array (of double) of size M containing the M marks of student i whose name is names[i], i being in the range [0, S − 1]. Notice that for each valid index i, marks[i][j] is one mark of student i, j being in the range [0, M − 1].  
  
**Supporting file**