2. Control structures: selection and repetition.

2.1.

Solve again problem **1.4** (solution of a system of linear equations in two variables), so that when the system is impossible or inconsistent (a system having infinite solutions) a message is shown to the user: "impossible system" or "inconsistent system".

2.2.

a) Write a program that reads 3 numbers from the keyboard and determines the largest and the smallest number.

b) Write a program that reads 3 numbers from the keyboard and writes them on the screen, in descending order.

c) Write a program that reads 3 positive numbers from the keyboard and determines if they can represent the length of the 3 sides of a triangle (tip: it is not possible to build a triangle if the sum of the 2 smallest lengths is smaller than the largest length). If any of the numbers is not positive the program must show an error message.

2.3.

Write a program that reads 2 integer numbers from the keyboard and tests whether their sum would produce overflow (the result would be greater than **INT_MAX**) or underflow (the result would be lower than **INT_MIN**). If this happens the program must show the message "sum overflow" or "sum underflow", otherwise it should show the result of the sum.

2.4.

The cost of transporting a certain merchandise is determined, depending on its weight, as follows: if the weight is less or equal to 500 grams the cost is 5 euros; if the weight is between 501 grams and 1000 grams, inclusive, the cost is equal to 5 euros plus 1.5 euros for each additional 100 grams or fraction above 500 grams; if the weight exceeds 1000 grams, the cost is 12.5 euros plus 5 euros for each additional 250 grams or fraction above 1000 grams. Write a program that, given the weight of a certain merchandise, determines the cost of its transportation.

2.5.

Write a program to determine the roots of a quadratic equation $Ax^2+Bx+C=0$, the coefficients ***A***, ***B*** and ***C*** being provided by the user. The program must indicate whether the equation has 2 different real roots, 2 equal real roots or 2 complex roots, and the respective root values , with 3 decimal places.

Example:

```
Solution of Ax^2 + Bx + C = 0
Insert the coefficients (A B C): 2.5 -1 16
The equation has 2 complex roots: 0.200+2.522i  and  0.200-2.522i
```

2.6.

Write a program to determine and write the amount that a depositor can withdraw from the bank, after **n** years of depositing an amount **q**, where **j**% is the annual interest rate. The values of **n**, **q** and **j** must be specified by the user. Assume that interest at the end of each year is accrued to the deposited amount.

2.7.

A number **n** is prime if it is divisible only by itself and by one.

a) Write a program that reads a number from the keyboard and determines if it is prime. Note: it is not necessary to test all divisors in the range **[2..n[**; it is enough to test divisors until the integer part of the square root of **n**.

b) Write a program that writes on the screen all the prime numbers lower than 1000.

c) Write a program that writes on the screen the first 100 prime numbers.

d) Write a program that determines the largest prime number that can be stored in a varible of type **unsigned long**.

2.8.

a) Write a program that displays on the screen a table of sines, cosines and tangents of the angles in the range [0..90] degrees, with intervals of 15 degrees, as shown below (note the particular case of the last line, corresponding to the 90 degree angle).

ang	sin	cos	tan
0	0.000000	1.000000	0.000000
15	0.258819	0.965926	0.267949
30	0.500000	0.866025	0.577350
45	0.707107	0.707107	1.000000
60	0.866025	0.500000	1.732051
75	0.965926	0.258819	3.732051
90	1.000000	0.000000	infinite

b) Change the program you developed in **a)** so that the range limits and the interval of the value of the angles in the table can be specified by the user (for example, if the range is [0..1] and the increment is of 0.1 degrees, the table for the angles of 0, 0.1, 0.2,…, and 1 degree should be displayed).

2.9.

A palindrome is a word, number, phrase, or other sequence of characters which reads the same backward as forward. For example, the following numbers are palindromes: 12321, 555, 45554 and 11611.

a) Write a program that reads a 3-digit integer and determines whether or not it is a palindrome (suggestion: use the division and module operators to separate the integer into its digits.

b) Generalize the program in **a)** in order to treat unsigned integers with a greater number of digits. Note: do not use *arrays* or *vectors* to store the digits.

2.10.

Write a program that reads an integer and breaks it down into prime factors (example: 20 = 2x2x5).

Notes:

- one way to solve this problem would be to start by dividing the number by the first prime number, 2, and continue dividing by 2 until you get non-zero remainder; then divide by the other prime numbers, 3, 5, 7, etc. until the only numbers left are prime numbers;
- an alternative way is to start by dividing the number by 2 and continue dividing by 2 until you get non-zero remainder; then do the same for 3, 4, 5, 6, 7, etc. until the dividend is equal to 1.

2.11.

Write a program to calculate the sum of the first **n** terms (**n** being input by the user) for each of the following series.

a) Series giving the value of the mathematical constant π:

$$4-\frac{4}{3}+\frac{4}{5}-\frac{4}{7}+\frac{4}{9}-\frac{4}{11}+\ldots$$

b) Series giving the value of the mathematical constant *e* :

$$1+\frac{1}{1!}+\frac{1}{2!}+\frac{1}{3!}+\ldots$$

(Tip: Calculate each term from the immediately preceding term)

c) Series giving the value of e^{-x} (com *x* real positivo previamente definido):

$$1-\frac{x}{1!}+\frac{x^2}{2!}-\frac{x^3}{3!}+\ldots$$

(Tip: Calculate each term from the immediately preceding term. Note: **x** must also be input by the user.)

2.12.

Repeat problem **2.11** so that the user can specify the precision with which he wants the result, that is, the maximum variation between the value of the sum of the series, between two consecutive iterations. Note that the variation can be either positive or negative.

2.13.

Write a program that reads a sequence of integer numbers, and determines and writes the sum, the mean, the standard deviation, the smallest and the largest of the numbers, in the following situations:

a) the length of the sequence is previously indicated by the user;

b) the end of the sequence is indicated by the value 0 (which is not considered to be part of the sequence); at the end, the program must indicate the length of the sequence;

c) the end of the sequence is indicated when the user types end of input (**CTRL-Z** in Windows; **CTRL-D** in Linux).

2.14.

a) The square root of a number **n** can be calculated in an approximate way using the following algorithm, due to Heron of Alexandria:
- start from an initial estimate of the root value, **rq**;
- calculate a new estimate, **rqn**, using the formula: **rqn = (rq + n / rq) / 2**;
- repeat this calculation using the **rqn** value as a new estimate of **rq**.

The following table illustrates the evolution of the square root calculation for the number **n** = 20, based on an initial estimate **rq** = 1.

rq	rqn	rqn^2	dif = n - rqn^2
1	10.500000	110.250000	-90.250000
10.50000	6.202381	38.469532	-18.469532
6.202381	4.713475	22.216844	-2.216844
4.713475	4.478314	20.055300	-0.055300
4.478314	4.472140	20.000001	-0.000039

In general, this algorithm converges quickly to a correct solution, even when the initial estimate is poor. Note that the difference **dif = n - rqn^2**, quickly evolves to zero, so one could use the value of **dif** as a stop criterion, that is, end the iterations when **dif** is less than a small number, **delta** (for example, **delta** = 0.00001). However, it is advisable to limit the number of iterations, repeating, in this case, the calculations until a value of **dif** less than **delta** is reached or a maximum number of iterations, **nMaxIter**, is reached.
Considering the presented example, if **delta** = 0.001 and **nMaxIter** = 3 the final result (**rqn**) would be the one calculated after the 3rd iteration because the maximum number of iterations specified has been reached, although the **delta** value is still greater than 0.001; but if **delta** = 0.001 and **nMaxIter** = 10, the calculation would end after the 5^{th} iteration because in this iteration the absolute value of **dif** is less than 0.001.
Write a program that reads the values of **n**, **delta** and **nMaxIter** and that calculates the square root of **n** using the algorithm described previously. Always use 1 as initial estimate of **rq**.

b) Modify the program in **a)** in order to present the result with the same number of decimal places used in specifying the **delta** value; for that you should find an algorithm to determine the number of decimal places. Also show the value returned by the **sqrt()** function from the C library and compare the values returned by this function and Heron's algorithm.

2.15.

Write a program to test if the user knows the multiplication tables. The program should generate 2 random numbers (between 2 and 9), present them to the user, and ask for the result of multiplying those numbers. After reading the user's answer, it should present an appropriate message taking into account the correctness of the answer and the time it took the user to give it: if the answer is wrong, the user should receive the message "Very Bad", if the answer is correct and took less than 5 seconds, the user should receive the message "Good", if you answer is correct but took between 5 seconds and 10 seconds (inclusive) the user should receive the message "Satisfactory", otherwise the user should receive the message "Insufficient".

2.16.

Write a program to emulate the operation of a basic calculator. In addition to performing the 4 fundamental algebraic operations, addition, subtraction, multiplication and division, the calculator must have a memory where the value shown on the display can be saved, using the **M** command. It must also be possible to clear the contents of the memory, using the **MC** command, add or subtract the current value on the display to the memory, using the **M+** and **M–** commands, or show the contents of the memory on the display using the **MR** command. The command to clear the contents of the display is **C**.