PROGRAMMING FUNDAMENTALS I – THEORETICAL WORK

The development of the work will consist of the design in pseudocode of the chosen program, its implementation in Java and the "test" of its correct performance by means of at least two different examples using different input data.

To obtain a minimum grade of 5 points, the program must follow the principles of structured programming and the concepts studied in the subject Programming Fundamentals I, and the use of object-oriented programming or advanced structures or classes is not recommended. The following aspects will be assessed negatively, which may mean that the work will not be passed:

- the use of global variables when it does not correspond,
- Incorrect modularization of the program,
- incorrect parameterization of the methods,
- incorrect use of parameters,
- the excessive size of the "main" method

In all programs the correct entry of data must be controlled, preventing the program from ending when incorrect values are entered, in that case the user will be given the opportunity to try again until the entry is correct. In other words, if, for example, a positive number has to be entered and the user enters a negative or zero value, the program will display a message warning of the error and provide the possibility for the user to try again, repeating this process as many times as necessary until the user enters the positive value. However, it is not necessary to control the cases in which the user enters a value of a different type of data from the one being requested. For example, if you are asked to enter a number and instead, a character is entered.

However, for any doubt or clarification about the statement and its approach to its resolution, you should contact your “theory” professor.

PROBLEMS

1. Quiniela

Each of the N members of a club fill out a 14-matche ticket. To do this, they must put the value '1', 'X' or '2' in each match, depending on the result they expect to occur, being '1' for the case in which the home team wins, 'X' for the tie and '2' for when the visitor wins. When all 14 matches have been played, the actual results of each of them are provided. Therefore, it is asked to make a program that allows to manage the bets of each of the members of the club, as well as the winning combination, to provide how many hits each one has obtained and who has been the one that has obtained more hits. To do this, the program will start by storing all the data, to then display a menu with different options for the user, who will decide when to end the program. This menu will consist of the following options:

I. Show a member's bets
II. Show all bets, along with the winning combination
III. Show the number of hits for a given person
IV. Show the person who has had the most hits
V. Show the complete information of the system: bets of each person, together with the winning combination, as well as the number of hits that each one of them has obtained.
VI. End of the programme

For the storage of information, the following considerations shall be taken into account:

- The number of members of the club (N) will be requested by keyboard and will be greater than 1.
- The 14 values corresponding to the bets of each person will be obtained by keyboard.

- The winning combination is obtained from a file containing 14 characters (values '1', 'X' or '2').

For options I and III, the program will ask for the identifier of the person whose bets or hits you want to know.

2. Race through the labyrinth

Two people compete to get out of a labyrinth, but they cannot move at the same time, they will always move one after the other, alternately, until at least one of the two arrives at the exit. For each movement they can only move one position, vertical or horizontal. The winner is the one who has travelled the shortest distance.

A program is requested that simulates the competition in an interactive way, for which it will be taken into account that the labyrinth will be represented as a board of NxM dimensions in which there will be a series of squares that represent the obstacles. The length of the path will be defined as the number of squares through which it has passed, so the winner will be the one whose path is formed by the least number of squares. To do this, the information representing the labyrinth will be loaded from a text file in which the first line contains the values of N and M, respectively; the second line will contain the starting square (row and column); the third line, the finish line; and the other lines, the positions in which there are obstacles. Once the labyrinth has been initialized, the names of the two runners will be asked for by the keyboard, and then the program will ask each one, depending on their turn, for the position to which they wish to move. To do so, the following restrictions will be taken into account:

- In the labyrinth there is at least one path from the entrance to the exit.
- Both players start from the same square, which represents the origin, and must reach the same destination.
- Luck will be drawn which of the two begins the game
- Both players cannot be in the same position at the same time.
- If a player cannot move from his position, he loses his turn.
- If a player attempts to move incorrectly across the board (occupied square or out of the labyrinth, or an obstacle), he will get a penalty consisting of one unplayed turn and an increase of 3 squares along the path.
- If a player attempts to gain access to a position he has previously been in, he will be penalized with an increase of 2 squares along the path.
- At any time, either of the two players can leave the game, regardless of the situation in which they find themselves, which will mean victory for their opponent and will be equivalent, to the one who leaves, that his path is N*M squares.
- The loser must pay the winner an amount E according to the difference in terms of squares between the two paths.
- For each turn, the situation of the labyrinth must be shown, with the path travelled by each player.

At the end of the program, the complete board will be shown, with the path covered by each of them, as well as information on the number of squares covered by each and the amount to be paid by the loser to the winner.

3. Sudoku

Sudoku is a mathematical game whose objective is to fill a grid of 9 × 9 cells (81 squares), divided into sub squares of 3 × 3, with figures from 1 to 9, taking into account that you should not repeat in the same row, column or sub grid, and that you start from N (N[1,81]) numbers already arranged in some of the cells.

It is necessary to implement a program that allows solving a sudoku in an interactive way. To do so, the program will start generating the initial sudoku, which will end when the board is complete. The generation of sudoku will be done from a text file that will contain, in the first line, the value of N and then each line will contain three numbers, corresponding, respectively, to the row, column and number of each of the N fixed values from which it starts. In addition, before and after each move the program will show on the screen the state of the board, always differentiating the "fixed" numbers from those filled by the player, and the user will have two options: insert a value in an empty cell or delete one of the values you have entered. In the first case, the program will ask by keyboard for the row and column of the position in which you want to enter a value and then it will ask for the number to assign; if it violates the sudoku's rules, the error will be warned and the sudoku will not change its state. To delete the value of a cell, the program will ask for the row and column in which the value you want to delete is located. The program should end when the sudoku is completed or when the player decides so, even if he has not completed it.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

4. Boats game

In the boats game there are 2 players, each with a board of N rows and M columns, in which each participant places his boats and each one tries to "sink" the opponent's boats. The boats' size is 1 (B1) or 2 (B2), occupying, respectively, one or two cells (in this case, they will be consecutive) and cannot be touched, i.e. two different boats cannot occupy consecutive cells vertically or horizontally. The game is that, alternatively, each player chooses a position (row and column) in which he thinks that there may be an opponent's boat. If there is no boat in the indicated position, the opponent will answer "water"; if there is a size 1 boat, the answer will be "sunk"; if there is a size 2 boat, "touched one of 2" or "sunk" if he has already "touched" both positions. Each player will store the "attacks" he has made to his opponent to try to hit the successive shots. The game ends when one of the participants succeeds in "sinking" all the boats of the other, who will be the winner.

It is necessary to implement a program that simulates the boats game. To do so, initially the values of N, M and B1 and B2 will be requested by keyboard, checking that all these values are positive. Each player will then place his B1 size-1 boats and his B2 size-2 boats in the positions on the board of his choice. The game then begins.

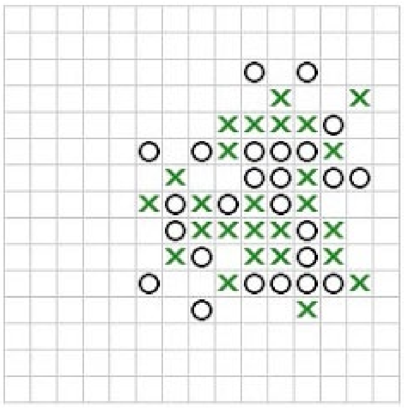
Example of a 4x8 board where 4 boats of size 1 and 3 of size 2 have been placed. It is observed that a boat of size 2 is not valid, it is crossed out, because it coincides in part of the side with another boat.

1				2	2		1
			1		~~2~~	~~2~~	
							2
	1		2	2			2

5. Mastermind game

Mastermind is a game for two players, which is played on a board with B chips white and N black, small, and C colors, somewhat larger. One of the players, the "master", chooses a number F of colored tiles (F=4 in the original game), and puts a secret code hidden from the other player. This one, taking colored chips from the same set, proposes a combination, which is answered by the "master" with black and/or white chips. The master places as many black tiles as the player has placed coloured tiles; and as many white tiles as the player has placed correctly coloured tiles, but in a different place. The game ends when the player gets the correct combination (i.e. a combination with four black pieces), or there are no more possible combinations (depending on the size, although there are usually 10 combinations).

It is necessary to implement a program that allows you to play the Mastermind game, considering that the computer plays the role of "master", and that the number of colored pieces of the combination is an F value that is read by the keyboard. To do this, the program will start by initializing the board, for which the whole values B, N and C will be read from a text file, followed by the identifiers of the C colors, each one on a line. It will then ask the user to indicate the value of F and then randomly define the combination of F colors that the player will have to guess. Then the game starts. After each move, the program has to show the values entered by the player, along with the amount of black and/or white chips corresponding to the move. When finished, the program must show the winning combination, as well as all the attempts made by the player, along with the answers made by the program to each of the attempts.



6. tic tac toe

The game is played on a NxN board and two players participate with (N^2/2)+1 pieces of two different types, one for each one. The objective of each player is to get before the other a row of **five** pieces placed on the same vertical, horizontal or diagonal line. Randomly one of the two players starts the game. Both players are playing in an alternative way. Each move consists in adding a new chip to the board. The game ends when a player gets a row of 5 chips or when the board is completed, and no more chips can be placed.

It is necessary to implement a program that allows two players to play 5 in line in an interactive way. To do this, the value of N will be requested by keyboard, as well as the names of the two players. Then, the program will decide, randomly, which of the two players will start the game, as well as the type of chip with which it will play. The program will then indicate, alternately, which of the two players will make his move, for which, after each move, will show the status of the board. It is necessary to control all possible errors such as inserting data on squares already occupied or squares outside the dimensions of the board. When the game ends, the state of the board must be saved in a text file, for which the value of N will be saved in the first line, and then each line will represent a row of the board.

7. The queen of chess

In chess, a queen "eats" (or covers) horizontally, vertically and diagonally.

For example, on the right-hand board, the `R' position queen can "eat" (or cover) in all squares marked with an 'x'.

x			x			x	
	x		x		x		
		x	x	x			
x	x	x	R	x	x	x	x
		x	x	x			
	x		x		x		
x			x			x	
			x				x

Implement a program that, given a NxN board, where N is a positive integer value that is entered by keyboard, the user is entering queens in the rows and columns you want until the whole board is covered by the power of the queens. To do this, the program will ask the user the row and column of the box in which you want to place a queen. Each time a queen is introduced into a valid board position, not covered by a previously entered queen, the queen's position will be marked with the value 'R' and all the squares covered by the queen will be marked with the value 'x', then showing the state of the board. If when inserting a queen into a square, it is already covered by another queen entered earlier (i.e. the square has the value 'x' or 'R') or falls outside the edges of the board, the program will display a message to the user indicating the situation and will do nothing. The program ends when the whole board is covered by the effect of the placed queens, showing the number of used queens.

Following the example of the previous board, if a queen is introduced in the position (1,7) the board would be as in the figure on the right.

x			x			x	x
x	x	x	x	x	x	x	**R**
		x	x	x		x	x
x	x	x	**R**	x	x	x	x
		x	x	x			x
	x		x		x		x
x		x	x			x	x
	x		x				x

If you then try to put a queen in the box (3,1), as this position is already "covered" (there is an 'x'), the program will not do anything and will show a message indicating that "the box (3,1) is already covered". Similarly, if you try to place a queen on the square (8,9), the message would indicate that "the square (8,9) is out of range". In both cases, the number of queens used is not increased.

x			x			x	x
x	x	x	x	x	x	x	**R**
		x	x	x		x	x
x	x	x	**R**	x	x	x	x
		x	x	x			x
	x		x		x		x
x		x	x			x	x
	x		x				x

If several queens are placed successively, for example, in (0,5), (7,4), (4,0) and (2,1), the board would be completely covered and the program would end. In this case, 6 queens would have been used.

x	x	x	x	x	**R**	x	x
x	x	x	x	x	x	x	**R**
x	**R**	x	x	x	x	x	x
x	x	x	**R**	x	x	x	x
R	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x
x	x	x	x	**R**	x	x	x

8. The knight of chess

In chess, a knight "eats" (or covers) in the form of an 'L' in all directions.

For example, on the right-hand board, the knight from position `C', can "eat" (or cover) in all squares marked with 'x'.

		x		x			
	x				x		
			C				
	x				x		
		x		x			

Implement a program that, given a NxN board, where N is a positive integer value that is entered by keyboard, the user will enter knights in the rows and columns you want until the whole board is covered by the power of the knights. To do this, the program will ask the user the row and column of the box in which you want to place a knight. Each time a knight is placed into a valid board position, not covered by a previously entered knight, the knight's position will be marked with the value 'C' and all squares covered by that horse will be marked with the value 'x', then showing the state of the board. If when entering a knight in a square, it is already covered by a previously entered square (i.e. the square has the value 'x' or 'C') or falls outside the edges of the board, the program will display a message to the user indicating the situation and will do nothing. The program ends when the whole board is covered by the effect of the placed knights, showing the number of knights used.

					x		
		x		x			C
	x				x		
			C			x	
	x				x		
		x		x			

If you then try to put a knight on the square (2,1), as this position is already "covered" (there is an 'x'), the program will not do anything and will display a message indicating that "the square (3,1) is already covered". Similarly, if you try to place a knight on the square (8,9), the message would indicate that "the square (8,9) is out of range". In both cases, the number of queens used is not increased.

					X		
		x		x			C
	x				x		
			C			x	
	x				x		
		x		x			

If several knights are placed successively, for example, in (2,2), (4,4), (0,7), etc., the ones shown in the figure, the board would be completely covered, and the program would end. In this case, 21 knights would have been used.

C	x	C	x	x	x	C	C
X	x	x	x	x	x	C	C
X	**x**	C	x	x	x	x	x
X	C	C	C	x	x	x	x
X	x	x	x	C	x	C	x
X	x	x	x	x	C	x	C
X	C	x	x	x	x	C	x
C	C	C	x	x	C	x	C

9. Diagonals of matrices

Given a square matrix of dimension MxM, the diagonal to the right of row f is defined and column c is defined like this:

Dd(f,c)={M[f+i][c+i] con i=0,1,2.. y (f+i)<M.length y (c+i)<M[0].length}

Only the diagonals to the right of the first row (in case of f=0) or of the first column (in case of c=0) can be obtained.

For example, given the matrix:

Row/column	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

The following diagonals could be obtained to the right:

Dd(0,0)={1,5,9}; Dd(1,0)={4,8}; Dd(2,0)=7; Dd(0,1)={2,6}; Dd(0,2)={3}

Implement a program that computes the diagonals to the right of a matrix whose data will be obtained from a file. The values will be stored in the matrix sequentially by rows, that is, first the first row, then the next row, etc., until the data in the file is empty or the matrix is completed. In the first case, if there are fewer numbers in the file than cells in the matrix, the rest of the elements of the matrix would be completed by 0's. Next, the program will show a menu that allows the following operations:

I. Show the sums of all diagonals.
II. Show the sum of a single diagonal, the row and the corresponding column would be requested by keyboard. In this case, the program must check that the entered data are correct (the row and/or the column are 0). In this case, the corresponding diagonal will also be shown.
III. Show the M matrix.
IV. Show all diagonals
V. Exit the program

For data storage, the M dimension of the matrix and the name and location of the file containing the data of the matrix shall be read from the keyboard. Then, the data of the file will be loaded to the matrix created taking into account that if the file has less data than components in the matrix, the matrix will be completed with 0's, and if it has more, only the first MxM will be loaded.

For example, if M=3 is entered and the file contains the values "0 1 2 3 4 5 6 7 8 9", the M array should be [[0 1 2] [3 4 5] [6 7 8]]; if the file contains the values "0 1 2 3", the M array will be [[0 1 2] [3 0 0] [0 0 0]].

10. Timetable

We want to develop a digital timetable of a course with the subjects taught in 6 time slots (three before break and three after break) throughout the 5 working days so that the user can consult it at will. To do this, a program must be implemented showing a menu with different options for the user, who will decide when to finish the program. This menu will consist of the following options:

- Create timetable
- Show the days and hours on which a particular subject is taught.
- Show the schedule of a given day
- Show full schedule
- Change the name of a subject of the timetable
- End the programme

For the initialization of the schedule will be taken into account:

- The data is read from a text file, in which the first line will contain the number of subjects and the rest of the lines will contain two values: an integer and, separated by a space, a text string, which will represent, respectively, the number of hours and the name of the subject.
- If the total number of hours for all the subjects is less than 30, an error message must be shown indicating to the user that the input file is not correct, and the program will end.
- Each subject has only one hour a day
- Subjects that have less than three hours per week will be taught after break, whenever possible.
- Subjects that have more than three hours per week will be taught before break, whenever possible.

In order to show the days and hours in which a specific subject is taught, the user will be asked to enter the name of the subject he wants to see. To do this, the names of the subjects that are part of the timetable will be first shown to the user.

In order to show the timetable of a specific day, the user will be asked to enter by keyboard the day of the week whose timetable he wishes to see. Afterwards, the program will show each hour of the day along with the subject.

To show the complete timetable, it will be done in the form of a table, but previously the user will be asked if he wants to visualize it horizontally or vertically. In the first case, the rows will correspond to the hours and the columns to the working days; in the second case, the rows will correspond to the days and the columns to the hours.

To change the name of a subject, the user will be prompted to enter first the name of the subject of the timetable you want to change, and then the new name.

11. Horror Film Marathon

We would like some information on the profits made during a horror film marathon which was held over the course of one night in cinemas consisting of 5 rooms and in which, 5 films were shown per room, all were 120-minute duration, one after the other, but obviously in different order. All tickets have the same price, regardless of the room and the film. To do this, a program must be implemented displaying a menu with different options for the user, who will decide when to finish the program. This menu will consist of the following options:

I. Create the tables with the spectators that each film has had in each room
II. To show the room that has had more spectators throughout the night.
III. Show films that have been screened in a given room
IV. Show total marathon spectators
V. Show the time with the highest number of viewers
VI. Show the full schedule of the marathon, along with the spectators of each screening.
VII. End the programme

For the creation of the table of spectators will be taken into account:

- The data is read from a text file, in which the first line will contain the number of films screened; the following lines will each contain positive numbers, separated by a space, followed by the title of the film that has achieved this number of spectators in each room (in order of appearance).
- If the number of movies is less than 5 (the number of rooms), an error message must be displayed indicating to the user that the input file is not correct and the program will end.
- If the number of films is greater than 5 (the number of room), the first 5 of the file will be chosen.
- Each film is screen only once per room.

In order to show the room with the highest number of spectators, the names of the films screened in that room will be shown on the screen, along with the spectators of each of them.

In order to show the films that have been screened in a given room, the user will be asked to enter the room whose information he wishes to know by keyboard. Afterwards, the program will show each time slot together with the screened film.

In order to show the time slot that has had the most viewers, the films screened in each cinema at that time will be shown on the screen, along with the number of viewers for each of them.

To show the complete schedule, it will be done by a table, but previously the user will be asked if he wants to visualize it horizontally or vertically. In the first case, the rows will correspond to the hours and the columns to the rooms; in the second case, the rows will correspond to the rooms and the columns to the time slots. In both cases, along with each film, the number of viewers obtained for each cinema in each time slot will be indicated between brackets.

12. Tapas contest jury

During the weekend there will be a tapas competition in which 10 bars of the city participate, each with a different tapa, which will be judged by a 5-people jury. Each member of the jury must vote in favour of 5 tapas and a punishment vote for one of them. In order to facilitate the task of the jury, we wish to draw up a programme that will be in charge of managing all the information, counting the votes cast and obtaining the winner. Thus, this program will consist of the following options:
I. Obtain from a text file the names of the tapas taking part in the competition.
II. Obtain from the keyboard the DNIs of the members of the jury.
III. Vote Management
IV. Show a menu to the user consisting of the following options:

a. Provide the name of the winning tapa, as well as those that have obtained the 2nd and 3rd position.

b. Provide the name of the punished tapa, i.e., the one with the most punishment votes.

c. Provide the tapa or tapas that have not obtained any vote, either favorable or against

d. Identify the member of the jury who has made a mistake when voting, either because he has cast less than 5 votes in favor or less than 1 punishment vote.

e. Identify if any member of the jury has not voted

V. Show the complete board with the results of the votes, except for the covers that did not get any votes.
VI. Finish the program

In order to read the data from the text file, the first line of the file will be considered to contain the number tapas participating and then their names in the remaining lines, each one in a different one. If the number of tapas is less than 10 (number of bars), an error message will be thrown, and the program will end. If the number of tapas is greater than 10 (number of bars), only the top 10 will be considered.
For the management of the votes it will be taken into account that, for each member of the jury, the programme will provide the participating tapas, as well as an identifier for each one of them. Next, the program will ask for their favorable votes, one by one, with the user indicating when he or she wishes to stop voting; then, the program will ask you if he/she wishes to cast your punishment vote, being the user who makes that decision and, if so, indicating the tapa that is voted.

13. Online cinema

An online cinema allows the user to store the premiere films and the opinions of film critics by using stars, 1 star being the minimum score and 5 the maximum score. To facilitate the task of the jury, we want to develop a program that manages all the information, counting the votes cast and obtaining the winner. Thus, this program will consist of the following options:

I. Read by keyboard the number N of films that the cinema has
II. Read by keyboard the titles of the N movies
III. Read by keyboard the M number of critics
IV. Each critic will be asked for their name, then the name of each film will be shown, and he/she will be asked for the score of each one.
V. Display a menu to the user consisting of the following options:

a. Show the movie with lower average

b. Show the movie with higher average

c. Show the critic with the lowest average score given to all the films (If there are several, show only one of them).

d. Show the film with the lowest individual score (not average) by any single critic (If there are several, show only one of them).

e. Show all movies with more than 4 starts on average

VI. Exit the program

At the end of the program, save in a text file the names of the films and the associated average scores. For this purpose, the name of the film will appear in one line and the average score in the next line.

14. Online betting system

An online betting system loads a card containg NxN numbers, in the interval [1,50]. There are 4 possible games:

- Major: The player will win if he selects 4 consecutive cells in such a way that the second is smaller than the first, the third is smaller than the second, and the fourth is smaller than the third.
- Minors: The player will win if he selects 4 consecutive cells in such a way that the second is bigger than the first, the third is bigger than the second, and the fourth is bigger than the third.
- Divisors: The player will win if he selects 4 consecutive squares in such a way that the second and first cells have a common integer divisor, the same with the second and third squares, and the third and fourth cells.
- Odd: The player will win if he selects 4 consecutive odd cells.

Implement a program that simulates the system, bearing in mind that at the beginning the player's card will be loaded from a text file whose first line will contain the value of N and for each of the remaining lines, the values of the card ordered by rows from left to right. Next, the program will offer the user a menu with the different games offered. When the game is over, it will show whether the player has won or lost and give them the option to play again or finish. To select the cells, the program will ask for the corresponding row and column where he/she wants to play. Each time the player selects a cell the system will check whether the conditions of the chosen game are met and, if so, will give the player the option to continue; if not, the chosen game will end.

15. Organizing the closet

In a closet with N shelves, M cans of various types (e.g. "milk", "tomato", "juice", etc.) are placed. Implement a program that, once the shelf is created, provides the user a menu with the following options:

I. Grouping: Starting from the left of the top shelf, all products of the same type will be placed in alphabetical order.
II. Replace an element from a shelf. To do this, the user will be asked to enter the shelf and position of the can to be replaced, as well as the name of the new product.
III. Replace all the elements of a shelf. To do this, the user will be asked to enter the shelf, as well as the name of the new product, replacing all the cans on that shelf with the new one.
IV. Replace the element with fewer units with the more abundant one. The most abundant element and the least abundant will be counted, and all the appearances of the least abundant of the shelf will be replaced by the most abundant.
V. Show the current state of a shelf.
VI. Show the complete closet.
VII. Exit the program. In this case, the status of the entire closet is stored in a text file.

The initial content of the closet will be loaded from a text file, where the first line will be the number of shelves N. Next, on each line there will be a product, which will be placed on the right starting with the bottom shelf, until each shelf is completed.

16. The invisible board

It is a game in which M players participate (M>=2 and M<=N/2) on a NxN board of integers (with N>3), which has values in nonconsecutive cells, one filled cell, another empty, alternately, according to the distribution of the following example:

Num1		Num2
	Num3	
Num4		Num5

The numbers placed will be "invisible", i.e. the players will never see them. The game consists in introducing values in the empty cells, so that, if the number is between the adjacent values, it will be inserted in the board and counted as a success for that player; otherwise, it will be counted as an miss and that cells will be left blank for the rest of the game. For example, if it is necessary to enter a number X in position [1, 0] (considering 0-origin), the value X must be in the range [Num2, Num3]. The turns between the players alternate consecutively, so that each player can only try to insert one value in each turn. The game ends when there are no more cells on which to insert values. The winner is the player hitting more, and there may be a tie between several players.

Implement a program that simulates the game. To do this, the program will load from a text file, both the size of the board (N) and its content, so that the first line will contain the value of N and then the rest of the numbers on the board. Then, the value of M will be read by the keyboard, indicating the number of players that will participate, along with their names. The program will randomly assign the order of the players' turns, which will be kept over the game. In each turn, the program will indicate the name of the player whose turn it is to play and will ask you to enter, by keyboard, the row and column of the square on which you want to assign a value, as well as the value you want to enter. The program will check that the row and column are correct, i.e. that they correspond to an empty cell; otherwise, the player loses his turn. At the end of the game, the system will show the final board state, with the values in each cell, the names of the players, the number of hits and misses of each player, the values entered by each player and the number of blank squares left on the board (misses).

17. Debate

You want to manage the debate between a number N (N>2) of users. To do this, a program will be made to read by the keyboard the number N of users and then read by the keyboard, also, the real names, along with a nickname for each of them. Then, the conversation will begin, showing a menu with the different options:

I. Speaking: In this case, the nickname of the user who wants to speak will be asked by keyboard, as well as the text you want to express (less than 256 characters). The content of the conversation will be saved in a text file in which the nickname of each participant appears, followed by a colon, along with the text emitted in quotation marks and ending with a full stop.

II. Viewing: The content of the conversation developed so far will be displayed.

III. Recovering chat: A complete conversation will be loaded from a text file. To do this, the program will ask you to enter the name of the text file containing the conversation.

IV. Ending conversation: In this case, the program also ends, showing the names of the users who actually participated in the conversation, along with the number of interventions carried out by each of them, as well as the total number of characters of the total of their interventions, indicating who has intervened the most and who has been the least, in terms of size (number of characters) of the total of their interventions.

18. Academic year qualifications

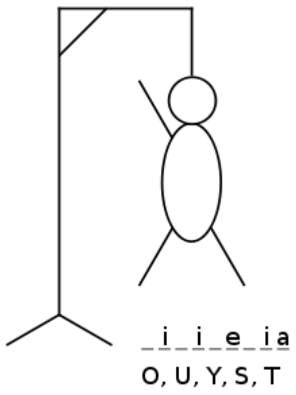
The N students enrolled in a course have received classes of M subjects, obtaining a final grade for each one of them. From these data, the tutor wants to obtain several descriptive statistics.

A program is requested capable of managing this information, bearing in mind that, firstly, the data will be loaded from three text files: the first will contain the number M of subjects and then their names, one per line; the second file will contain the number N of students and then their ID numbers, one per line, and the third will contain at least N*M real numbers representing the grades for each student. Once the system is initialized, the program will provide the user with a menu with the following options:

I. Subjects management: the information about a subject will be asked to be entered by the user, and then another menu will be shown with three options to offer the calculation, respectively, of the average, median and mode; the ID (or IDs) corresponding to the highest grade; and the ID or IDs of the lowest grade.
II. Students management: The user will be asked to enter the ID of the student whose information he wishes to obtain and then another menu will be displayed with four options to offer him the calculation, respectively, of the subject in which he has obtained the highest grade; the subject in which he has obtained the lowest grade; the grades in all subjects; and the global average of all subjects.
III. Mixed management: a menu with several options will be shown to the user, respectively, to show the subject that has obtained the highest number of subjects passed; the students' IDs that have obtained the highest grade in the course, indicating the subject or subjects in which they have obtained the highest grade; the total number of subjects passed; and the average of the course for all subjects.
IV. Show the complete system information
V. Exit the program

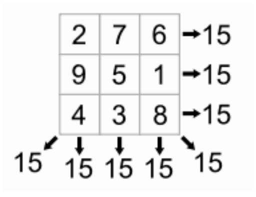
19. The hanged man

The hanged man is a game for two players. One of them, the guard, thinks of a word or sentence and the other, the executed, tries to guess it. As a clue, the guard uses a row of dashes, which represents the word to be guessed, with a dash by letter or number, in addition to the category to which it belongs. If the executed person suggests a letter or number that appears in the word, the guard writes it in all its correct positions. If the suggested letter or number does not appear in the word, the guard adds an element from the figure of a hanged man as a way of counting errors. The game ends when the executed person guesses the complete word correctly or when the guard completes the hanging diagram.



It is requested to implement a program that allows playing this game in an interactive way with all words from the file. To do this, the program will play the role of the guard and the words or phrases will be obtained from a text file, in which each line will represent a word or phrase and, the next, the category to which it belongs. The program does not need to draw the hanged man, but it does need to use some system to let the player know how many times the user has missed so far, and how many more time he can miss before losing.

20. Magic Square game



2	7	6	→15
9	5	1	→15
4	3	8	→15
↙15 ↓15	↓15	↓15	↘15

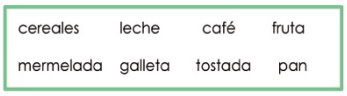
A magic square consists of a NxN board in which the numbers from 1 to N^2 are placed without repeating, so that all the main rows, columns and diagonals add up to the same amount.

The game, for J players, is that everyone should try to build a magic square, but starting from a random placement of the numbers from 1 to N^2. For this, the only movement allowed is the exchange of values between two squares. At the beginning of the game, the participation order between the players is established and will be maintained throughout the game. Whoever manages to build a magic square first wins.

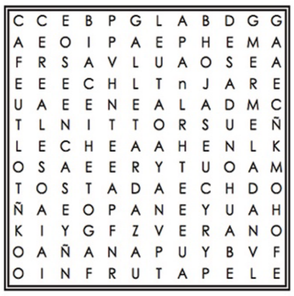
Implement this game for the case N=3 and a number of players, J, which will be read by the keyboard. The turns will be assigned randomly by the program. The initial assignment of the numbers will be obtained from a text file and will be placed in rows, from left to right and from top to bottom. The program will end when one of them obtains a magic square, in that case the mistakes made by each of the players will be shown. A mistake will be considered when a row, column or diagonal that does not add up to the amount solved by the square. On the other hand, if a player decides to stop playing the game, the number of mistakes will be the maximum, that is, to 8. In addition, before and after each turn the corresponding player's board will be shown; at the end of the program, the boards of each player will be shown, pointing out the successes of each board.

21. Crosswords

This hobby consists of finding a series of words within a board filled with letters; words can be "hidden" vertically, horizontally, or diagonally, in any sense.



cereales	leche	café	fruta
mermelada	galleta	tostada	pan



C	C	E	B	P	G	L	A	B	D	G	G
A	E	O	I	P	A	E	P	H	E	M	A
F	R	S	A	V	L	U	A	O	S	E	A
E	E	E	C	H	L	T	n	J	A	R	E
U	A	E	E	N	E	A	L	A	D	M	C
T	L	N	I	T	T	O	R	S	U	E	Ñ
L	E	C	H	E	A	A	H	E	N	L	K
O	S	A	E	E	R	Y	T	U	O	A	M
T	O	S	T	A	D	A	E	C	H	D	O
Ñ	A	E	O	P	A	N	E	Y	U	A	H
K	I	Y	G	F	Z	V	E	R	A	N	O
O	A	Ñ	A	N	A	P	U	Y	B	V	F
O	I	N	F	R	U	T	A	P	E	L	E

Implement a program that allows to solve a crossword in an interactive way. To do this, the program will ask for the matrix dimensions by the keyboard and then load the content of the matrix from a text file containing the characters that define the crossword. Then, it will show the words that the player has to find, which will be obtained from another text file and the game will start. To do this, when the player has placed a word on the board, he must indicate the positions in which each character is found. Each time the user enters a position, the program will check that it is "going well"; if at any time, the character on the board does not match the word, the program will give you two options: enter the correct position or start the search again. If when the player finishes entering all the positions, the word found coincides with one of the ones you must search, the program will display the word and the position in the matrix where it starts and will delete it from the list of words you must find. At any time, the user can stop playing, in that case the program will provide the list of words found.