4 February 2015 -- Computer Architectures –- part 1/2

Matr, Last Name, First Name ....

One of the typical family-with-children amusements during Christmas Vacations is to play "The Game Of The Goose" i.e. a racing board game where the goal is, by starting from cell 0, to be the first reaching the last cell of the race (cell number N). This is achieved by having the players, by turn, throwing a single die and advancing forward their placeholder by the number of cells as shown by the die. To make the game a bit more intriguing some cells are marked with awards and penalties to be executed by players landing on these cells.

It is requested to write a 8086 assembly program to allow always M=3 players (numbered 1, 2 and 3) to play the racing board game called "who graduates first?" (WGF). WGF has N=61 cells labelled from 0 (starting point) to 60 (end of the race), i.e. where the cells of the race are stored in the array WGF_RACE DB N DUP(?). For each cell of WGF_RACE at any phase of the game, it is stored the code of the corresponding award/penalty in the lower 4 bits (with code 0000 for normal, i.e. no award/no penalty cell), with all the upper 4 bits always set to 0000.

The awards and penalties are identified by codes as follows:

0001	double the number shown by the die, i.e. move forward again by the number shown by the die
0010	go forward by two cells
0011	throw the die once more and go forward (no penalty/award in the landing cell)
0100	miss a turn of die throwing
0101	go backwards by two cells
0110	throw the die once more and go backward (no penalty/award in the landing cell)
0111	say loudly "I love to play this crazy game and I am a liar!!!"

Students can assume that the awards and penalties are already decided and available both in number and type, before the game starts. Two very important characteristics of WGF are:

- once a player reaches a cell because of a die throwing he/she has to be subject to the award/penalty of that cell;
- while going forward or backward because of an award or a penalty, the corresponding award/penalty of the destination cell has not to be taken into account (and implemented).

The game starts by asking player 1 to throw the die (program is not required to draw the random value of the die) and enter the number. Then the program computes the landing cell and displays the numerical code (or, better, the full information) of the award/penalty/0000. If there is an award/penalty the program implements it and, before passing to the next player, it displays the name of the player and its current position. Then the program does the same with the next player and so on until the end of the game, when the winner is announced. The program clearly takes into account the additional actions such as the need to throw again the die or to rest for a turn. Example (for sake of saving printed space here, program's actions are condensed on single lines; in the final running version they can be on multiple lines):
Player 1 please throw the die and enter the number (1-6)
→ 3
cell 3 has an award: throw again the die (you will go forward)
→ 4
Player 1 is in cell 7; Player 2 please throw the die and enter the number (1-6)
→ 6
cell 6 has a penalty: miss one turn; Player 2 is in cell 6; Player 3 please throw the die and enter the number (1-6)
→ 4
cell 4 has no penalty/award; Player 3 is in cell 4; Player 1 please throw the die and enter the number (1-6)
→ 5
cell 12 has no penalty/award; Player 1 is in cell 12; Player 3 please throw the die and enter the number (1-6)
(player 2 is waiting as missed a turn)
→ 1
cell 5 has a penalty: throw again the die (you will go backward)
→ 6
Player 3 is in cell 0; Player 1 please throw the die and enter the number (1-6)
(player 3 cannot go lower than cell 0; he/she was in 5 and moved backward by 6 but stopped at cell 0)
→ 1
cell 13 has no penalty/award; Player 1 is in cell 13; Player 2 please throw the die and enter the number (1-6)
… *(and so on)*

4 February 2015 -- Computer Architectures –- part 1/2

Matr, Last Name, First Name ....

- **Only one (mandatory) of these 3 Items 1**; program has to comply with description and requirements above:
    - Item 1A: The winner is the first who reaches or surpasses the last cell of the race.
    - Item 1B: The winner is the first exactly landing to the last cell of the race. If a Player achieves a too much high value overflowing the last cell, he/she does not move from her/his last position.
    - Item 1C: The winner is the first exactly landing to the last cell of the race. If a Player achieves a too much high value overflowing the last cell, he/she reaches the final cell and continues counting backward the remaining steps, as shown by the die. Eg: position=58, die=6. Two steps forward let to reach the final cell 60; as four more steps have to be walked they are backward and the final destination is position=56.
- Item 2: to compute the following statistics: number of "1", "2",…"6" dice throws, the net balance of awards/penalties steps gained (for "miss one turn" assume a conventional loss of 4 steps) for each player, the number of die throws of each player once the game has ended.
- Item 3: to write an additional procedure (with global parameters) to double check if the final destination cell of a player (i.e. after all awards and penalties have been implemented) is already occupied by another player; if it is, then the position is decremented by one unit and the test is repeated until no busy cell is occupied. Example: Pl.1 in 15, Pl.2 in 14 and Pl.3 landing in 15… as 15 is occupied the new position becomes 15-1=14 which is again occupied and therefore the last position where to host Pl.3 is 14-1=13.

The points related to writing correct and completed items (as uncompleted items will not be evaluated) are:
- Item 1A: 21 points; Item 1B: 23 points; Item 1C: 24 points
- Item 2: 1 point for one statistics, 2.5 points for two, 5 points for all the three statistics
- Item 3: 6 points

Please consider that a maximum of 33 points can be accounted here; larger values will be "cut" to 33.

HINTS
- Please, if possible, avoid using nested if-then-else to implement awards/penalties depending on their code, but instead use a more general jump table.

REQUIREMENTS (SHARP)
- **Please choose and solve only one "Item 1" (either A, B or C).**
- It is not required to provide the optimal (shortest, most efficient, fastest) solution, but a working and clear one.
- It is required to write at class time a short and clear explanation of the algorithm used.
- It is required to write at class time significant comments to the instructions.
- The input-output part is not necessary in the class-developed solution, but its implementation is mandatory to be discussed at oral exam.
- Minimum score to "pass" this part is 15 (to be averaged with second part and to yield a value at least 18)

REQUIREMENTS ON THE I/O PART TO BE DONE AT HOME
- The database has to be defined and initialized inside the code
- All inputs and outputs should be in readable ASCII form (no binary is permitted).

Please use carbon copy ONLY (NO PICTURES ARE ALLOWED) and retain one copy for home implementation and debug. At the end of the exam please give to professors all the sheets of your solution. Missing or late sheet will not be evaluated. Please provide your classroom submitted solution with several explanatory and significant comments. Please remember that only what has been developed at class time can and will be evaluated at oral time and that it is necessary to write the instructions of the program and not just the description of the algorithm.
When coming to oral discussion, please clearly mark in red on your "classroom" copy, all modifications. Please also provide an error-free and running release of the solution, as well as with its printed list of instructions. Please consider that the above are necessary but not sufficient requirements to success the exam, since the final evaluation will be based on a number of parameters.
FAILURE TO ACCOMPLISH ALL THE ABOVE NECESSARY REQUIREMENTS WILL CAUSE NO-QUESTION-ASKED AND IMMEDIATE REJECTION.