15 June 2015 -- Computer Architectures –- part 1/2

Matr, Last Name, First Name **……**

Members enrolled in a Frequent Flyers' program (FFP) receive bonus rewards based on the miles flown, their status, and their class of booking according to the following rules:

Class of flight (K) and multiplier (X)

Class	Multiplier		Class	Multiplier		Class	Multiplier		Class	Multiplier
A	0.25		B	0.5		C	1		D	1
E	1.25		F	1.5		G	2		H	2

Status (S) of frequent flyer and bonus (B)

Status	Bonus		Status	Bonus		Status	Bonus		Status	Bonus
0	0		1	25%		2	25%		3	50%

Thresholds (T) in miles to change status of frequent flyer

Status	Threshold		Status	Threshold		Status	Threshold		Status	Threshold
0	0		1	3000		2	10000		3	40000

- Assuming that for one flight the nominal amount of miles is M, then the accumulated miles for that flight are computed as follows:
  - Flown miles weighted by the class = M * X; Reward miles weighted by the status = M * B
  - Accumulated miles for flight = flown miles weighted by class + reward miles weighted by status
- Each month all members of the FFP receive their account balance based on their previous status, miles accumulated in the month, current balance and their new status;
- The status of a member of the FFP can only change at the end on one month;
- It is assumed that there exist two databases, one with members' information and the other with all flights flown by members in one month, including the corresponding miles.

It is necessary to write an assembly 8086 program producing the account balance for each member and modifying the database with members' information accordingly. For example:

Members' information database

Member Number	Status	Miles accumulated		Member Number	Status	Miles accumulated
200	0	100		201	2	10005
202	0	0		203	1	3800

Flights flown by members

Member Number	Flight code	Class (C)	Miles (M)		Member Number	Flight code	Class (K)	Miles (M)
201	123	A	999		203	33	B	4023
203	34	B	833		202	4	H	9403

- For member 200 we do not have flights and therefore nothing changes
- For member 201 we have one flight (123) generating the following update
  - Accumulated miles for flight 123 = M * X + M * B = 999 * 0.25 + 999 * 0 = 249.
  - The status does not change and number of miles accumulated for member 201 is 10005 (previous value) + 249 = 10254
- For member 202 we have one flight (4) generating the update = M * X + M * B = 9403 * 2 + 9403 * 0 = 18806; status changes in 2 and number of accumulated miles is 0 + 18806 = 18806
- For member 203 we have two flights (33 and 34); the first generates 4023 * 0.5 + 4023 * 0.25 = 2011 + 1005 = 3016, while the second generates 833 * 0.5 + 833 * 0.25 = 416 + 208 = 624; the monthly accumulated miles is 3016 +624 = 3640 which leads to overall miles 3800+3640 =7440, thus making the status to remain 1.

Assumptions:

- There are only 8 classes of booking and 4 members statuses (see tables above)
- There are only 4 members to manage/track, numbered 200, 201, 202, 203, whose data are stored in the array FFM DB 4*4 DUP (?), according to the format (first entry is for member 200, second for 201 and so on)
  <status> <accumulated miles>
  8 bits 24 bits
- Flights of the month are stored in an array of at most 20 lines, FOM DB 30*5 DUP (?), each one in the format
  <member number> <flight code> <class> <flown miles> **(this flown miles value is not larger than 13000)**
  8 bits 8 bits 8 bits 16 bits

15 June 2015 -- Computer Architectures –- part 1/2

Matr, Last Name, First Name **……**

The 8086 assembly program should process the array FOM, display on the screen the changes (like in above) and display the updated contents of FFM (including the update on accumulated miles and the new status for each member) For simplicity, the user will provide the update through the keyboard at runtime. The final program will have to loop and a menu has to be provided to the user with the following options (the initial FFM is initialized in the program):

1) Fill the array FOM (member number, flight code, class, miles flown)
2) Compute and display the new parameters for all members based on FOM, including the details of variations (i.e. new miles overall gained and the status elevation, if any)
3) Reset all databases
4) Exit the program

IT IS FORBIDDENT TO USE THE MUL/DIV; ALL MULs/DIVs SHOULD BE DONE BY SHIFTS

- **(mandatory)** Item 1: the program neglects the status information of the members, but only computes and updates the flown miles according to the flight class and not the status extra bonus
- Item 2: the program considers also the status information of members, but does not check status elevation
- Item 3: the program checks and implements also the potential status elevation of members
- Item 4: information about miles accumulation is computed and displayed flight by flight (**TO GET POINTS, STUDENTS HAVE TO WRITE THE FRAMEWORK OF THE OUTPUT PART AT CLASS TIME, EXCLUDING THE DETAIL OF THE PROCEDURES FOR BINARY TO ASCII CONVERSION**)

The points related to writing correct and completed items (as uncompleted items will not be evaluated) are:

- Item 1: 24 points
- Item 2: 4 points
- Item 3: 4 points
- Item 4: 3 points

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

HINTS (observe that)

- it is not necessary to compute the values to maximum precision (truncation is fine)
- flown miles is not larger than 13000; this implies that the max number of miles generated by a single flight, to be added to the account is still a number that can be represented on 16 bits
- the number of accumulated miles is on 24 bits, while the number of generated miles for a flight is on 16 bits
- the comparisons to check a status elevation should be done between one value on 24 bits (accumulated miles) and another value sometimes on 16 bits and sometimes on 24 bits

REQUIREMENTS (SHARP)

- 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.