29 January 2018 -- Computer Architectures -- part 1/2

Matr, Last Name, First Name ....

January is a sale period and everyone is looking for the best offers and discounts. At Poly-MultipleDiscount (PM in the following) a huge multi-option-sale is running. During the promotional sale, each customer can buy no more than 4 products and, upon choosing only one option among these offered below, finally pay the due amount accordingly.

- **A**: pay the most expensive product at its full price and the least expensive with 50% of discount (minimum purchase = 2 products);
- **B**: pay the most expensive product with a surcharge of 25% (i.e. at 125% of its price) and get the least expensive for free, i.e. 100% of discount (minimum purchase = 2 products);
- **C**: pay the most expensive product with 12.5% of discount and the least expensive with 25% of discount (minimum purchase = 2 products);
- **D**: pay the most expensive product with 6.25% of discount, the second most expensive with 12.5% and the least expensive with 25% of discount (minimum purchase = 3 products);
- **E**: get the flat discount of 12.5% over the full purchase (minimum purchase = 4 products).

The following cases apply, depending on the number of objects purchased:

- In case of purchase of only one product, no discount option can/will be chosen.
- In case of two products, the customer has to declare which of the options A, B or C has been chosen.
- In case of three products, the customer has to declare which of the options A, B, C or D has been chosen.
- In case of four products, the customer has to declare which of the options A, B, C, D or E has been chosen; in this case, the customer has NOT the possibility to split the purchase into two sub-purchases of 2-products each.

Additional important information:

- The prices of the products on sale at PM, are stored in an array of 50 entries, each one on 16 bits (in pure binary) OBJ-PRICES DW 50 DUP (?) ; this array is available to the programmer.
- The number of products on sale (i.e., the number of valid cells of OBJ-PRICES) is stored in N DW ? and it is guaranteed that 4 <= N <=50; N is available to the programmer.
- The price of each product is given in cents and belongs to range [1000, 15000] (i.e. from 10 to 150 euro); this implies that the maximum amount spent is 4*15000 cents = 60000 cents, which can be represented on 16 bits.
- The products and their prices are sorted in the array OBJ-PRICES from the highest to the lowest.
- The products are referred by their position number in the array OBJ-PRICES; e.g.: item "0", item "4"...
- It is not necessary for the customers to buy different products; a customer can buy any combination of products, provided that they are no more than 4 (e.g.: two products "1", one product "2" and one "7").
- While computing the discounts (and/or discounted prices) the truncation errors have not to be managed and can be simply neglected. For example, if a product costs 1005 and 50% discount is applied, then it is equally correct to consider the discounted price as 503 (obtained as full price – discount = 1005 – 502 = 503) or 502 (obtained as full price * (1-discount percentage) = 1005 * 50% = 502).

It is requested to write a 8086 assembly program supporting the customer by providing the information about the final price paid and total discount amount, for the applicable discount options compliant with her/his purchase (one, two, three or four items). N and the sorted array OBJ-PRICES are assumed to be available to the programmer. The program starts by receiving in input both N_BUY (i.e. the number of products purchased by the customer) and the codes of the products that have been purchased. Then, the program computes & displays the values as requested by the item below that has been solved. Only fully completed items will be considered and please solve only one among 1, 2, 3 and 4.

- Item 1: POINTS → 21. Assume that the customer has bought **two** products only. Upon receiving in input the codes of the products purchased, compute (and print) the final price to be paid and the discount ONLY for the option **A**.
- Item 2: POINTS → 25. Assume that the customer has bought **one or two** products. Upon receiving in input the codes of the two products purchased, compute (and print) the final price to be paid and the discount for each of the options **A and B**. Clearly, the no-discount case, i.e., when only one product has been purchased, should be handled.
- Item 3: POINTS → 28. Assume that the customer has bought from **one to three** products. Upon receiving in input the codes of the purchased products, compute (and print) the final price to be paid and the discount for each of the options **A, B, and D**. Clearly, the no-discount case when only one product has been purchased, should be handled.
- Item 4: POINTS → 32. Assume that the customer has bought from **one to four** products. Upon receiving in input the codes of the purchased products, compute (and print) the final price to be paid and the discount for each of the options **A, B, D and E**. The no-discount case, when only one product has been purchased, should be handled.

Bonus Item ALPHA: Compute and display the percentage of discount (only if positive) of the full purchase, for all the options of the Item that was solved; POINTS → +2 if you solved Item 1 or 2; POINTS → +3 for Items 3 or 4.

12 II 14.30 – 18 lunedì

10 I 14.30 – 17.30 giovedì 17.30 – 18 lab.

29 January 2018 -- Computer Architectures -- part 1/2

Matr, Last Name, First Name .....

Bonus Item BETA: Assume that the customer has bought **two or more** products. Upon receiving in input the codes of the products purchased, compute (and print) the final price to be paid and the discount for option **C**. POINTS → +2.

Examples: N=11 and array OBJ-PRICES given below

Prod. #	0	1	2	3	4	5	6	7	8	9	10	11
Price	15000	13000	13000	10000	9999	7001	5000	4999	2099	1999	1000	1000

Purchase of products 0, 4, 4, 7. For option A we have: 15000+9999+9999+4999*50% = 37497 to be paid; full price = 15000+9999+9999+4999 = 39997; discount = 39997-37497= 2500; % of discount = (2500)*100/39997 = 6%, ...

	prod #0	prod #4	prod #4	prod #7	total	discount	% disc
cost	15000	9999	9999	4999	39997		
A	15000	9999	9999	2499	37497	2500	6%
B	18750	9999	9999	0	38748	1249	3%
C	13125	9999	9999	3749	36872	3125	7%
D	14062	8749	9999	3749	36559	3438	8%
E	13125	8749	8749	4374	34997	5000	12%

	#4	#5	#10	#11	total	disc.	%ds
cost	9999	7001	1000	1000	19000		
A	9999	7001	1000	500	18500	500	2%
B	12498	7001	1000	0	20499	-1499	n.a.
C	8749	7001	1000	750	17500	1500	7%
D	9374	6125	1000	750	17249	1751	9%
E	8749	6125	875	875	16624	2376	12%

	#1	#2	#3	total	disc	%ds
cost	13000	13000	10000	36000		
A	13000	13000	5000	31000	5000	13%
B	16250	13000	0	29250	6750	18%
C	11375	13000	7500	31875	4125	11%
D	12187	11375	7500	31062	4938	13%

HINTS (observe that)

- **The identification of the most and least expensive products is easy, as OBJ-PRICES is sorted.**
- **For the computation of Bonus-Alpha, it is better to compute first (discount*100) as a 16x16 bits multiplication, immediately followed by the 32/16 bits division by the total cost. As the result is a percentage, by definition it is less than 100 and therefore it is found in the least significant byte of AX.**

IMPORTANT NOTES AND REQUIREMENTS (SHARP)

- It is not required to provide the/an optimal solution, but a working and clear one using all information provided.
- It is required to write at class time a short & clear explanation of the algorithm and significant instruction comments.
- Input-output is not necessary in class-developed solution, but its implementation is mandatory for the oral exam.
- Minimum score to "pass" this part is 15 (to be averaged with second part and to yield a value at least 18)
- **To avoid misunderstandings, please consider that, as in the previous calls of the last 6 years (at least) the final score reflects the overall evaluation of the code, i.e., fatal errors, such as division by zero (etc) make it impossible to reach 30 or larger scores. Specifically, at oral exam, students will request the evaluation of some or all the parts that they have solved; prior proceeding to the correction, the points of the parts to be corrected will be added up and bounded to max 35. The final score, after the correction of students' requested items, will be "cut off" to 32.**

REQUIREMENTS ON THE I/O PART TO BE DONE AT HOME

- The databases (if any) have to be defined and initialized inside the code; in this case, all data related to the products purchased and their number have to be input from the keyboard **(i.e. NOT stored in the array/variables)**
- 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.