Course:	**COAL Lab**	**Code:**	**EL213**
Program:	**BS (Computer Science)**	**Semester:**	**Fall 2018**
Duration:	**120 minutes**	**T. Marks:**	**40**
Date:	**Friday 26-10-2018**	**Weight**	**20**
Section:	**All**	**Page(s):**	**1**
Exam:	**Lab midterm**		

Instructions/Notes:

- Use of the internet, notes, codes, lab manuals, and flash drives is strictly prohibited.
- You are only allowed to use the soft copy of book.
- Plagiarism will result in **F** grade in lab.
- Submission path: Section-X (here X will be your section A or B or C) \\sandata\xeon\Fall 2018\Shakeel Zafar\MID COAL\Section-X\Q1 or Q2
- Code must be **indented properly**, failure to comply will incur a penalty.

Question # 1: 20 marks

Xorq has invented an encryption algorithm which uses bitwise XOR operations extensively. This encryption algorithm uses a sequence of nonnegative integers $x = [x[1], x[2], x[3], x[4], \ldots . x[n]]$ as its key. To implement this algorithm efficiently, *Xorq* needs to find maximum value of $(a \oplus x[j])$for given integers *a, l and r,* such that $(l \leq j \leq r)$. Help *Xorq* implement this as an assembly subroutine.

The subroutine takes the following parameters through the stack: address of array *x, a, l, r* and returns the maximum of *(x[j] ⊕a)* in DX register. (Here '⊕' stands for xor operation)

For example, x= {3,5,9}, *a=4*, *l=0* and *r=2*. We test each *x[j]* for all values of *j* between *x[l]* and *x[r]* inclusive:

j	x[j]	x[j] ⊕4
0	3	7
1	5	1
2	9	13

The maximum value is 13.

Detail of array x and its encryption

Note: *r* will always be greater than *l, and x[l] and x[r] will always be a part of the array.* The numbers in *array x* will range from 0-65535.

Question # 2: 20 marks

You are given two arrays, array1 and array2 along with their sizes, size1 and size2, respectively. Both arrays are sorted already, but array1 is sorted ascending order while array2 is sorted in descending order. You have to **merge** the two arrays **in place, so** that the resulting array looks sorted in **descending** order and contains elements from array1 and array2.
For this, you cannot use **any sorting** algorithm. You cannot use the **stack** for sorting/merging. You **cannot** use any **extra memory** location/variable. Use **registers**.

Sample input/output

State of data variables **before** running the code:	State of data variables **after** running the code:
array1: db 1,2,3,4,5,6,7 array2: db 9,6,4,1 size1: db 7 size2: db 4	array1: db 9,7,6,6,5,4,4 array2: db 3,2,1,1 size1: db 7 size2: db 4