APS Assignment 3

Q1: Implementation of suffix array
Implement a Suffix Array that is capable of performing following operations on Strings in a most efficient way.

1. Given a string S print its minimum lexicographic rotation. O(nlogn)

2. Given an integer K, print the length of the longest substring that appears in the text at least K times.If no such substring exist, print -1. O(nlogn)

3. Given a strings S determine its longest substring that is also a palindrome. In the case of multiple solutions, print the lexicographically smallest palindrome. O(nlogn)

Evaluation Criteria:- You will be given a large string S (length <= 10^5), and one of the above cases. Print the corresponding output.

String consist of either Lower/Upper Case Alphabet and Numeric digits.

Note: For each subpart implement a different code. Submit it as Q1a_rollnumber.cpp, Q1b_rollnumber.cpp, Q1c_rollnumber.cpp

Example :
S = “dcabca”
1. All possible rotation are “dcabca” , “cabcad” , “abcadc” , “bcadca” , “cadcab” , “adcabc” . Among all lexicographically minimum is “abcadc” .

2. If K=2 than since “a” is the only substring that appears twice and its length is 1, so the answer is 1.

3. Since only length 1 substring are palindromic, and among them “a” is lexicographically smallest, hence answer is “a”.

Q2: Trie Implementation

Given an array **A** of **N** numbers, you will be given **q** queries.
Each query will contain a single integer **x.** You have to find then maximum xor of **x** from any number in **A.**
Constraints :
1 <= N, q <= 10^5
1 <= A[i] <= 10^{12}

Example :
A = {1, 2, 3}
x = 4

Maximum xor of x is with 3, therefore answer is 4 xor 3 = 7
Note : Submit it as Q2_rollnumber.cpp

Q3: External Sorting

Problem Statement: External Sorting is a class of algorithms used to deal with massive amounts of data that do not fit in memory. The questions aim at implementing one such type: K-Way merge sort algorithm to sort a very large array. This algorithm is a perfect example of the use of divide and conquer where with limited resources large problems are tackled by breaking the problem space into small computable subspaces and then operations are done on them.

Input Constraints:1. A file containing a large unsorted list of integers (Will not fit in your usual Laptop RAM).
2.Do not use any in-built data structures.

Output : A file containing non-Descending sorted list of the given integers
Languages allowed : C, C++, Python Evaluation parameters :

1. Time and Space Complexity of the algorithm
2. Efficient use of Data-Structures

Submission format: Your code should take two arguments. First is the name of input file. Second is name of output file.

Example Format:
If your input file is at ./data/input.txt And if you need your output file at ./data/ named output.txt
For c++, code should be of format ROLL_NO_3.cpp compiled file should accept two arguments ./a.out “./data/input.txt” “./data/output.txt”
For Java, code should be named ROLL_NO_3.java Compiled file will be ROLL_NO_3 java ROLL_NO_3 “./data/input.txt” “./data/output.txt”
For python, code should be named ROLL_NO_3.py python ROLL_NO_3.py “./data/input.txt” “./data/output.txt”