CS 102
ASSIGNMENT 1
Deadline - April 7, 23:59 hrs

A. Please submit the C/C++ codes for solving the following questions. Please put suitable comments in your codes so that they are easy to understand.

B. For questions 1-6, please use the data structures mentioned before each question within [ ]. For questions 7 and 8, you may use any suitable data structure.

C. Please try to optimize your algorithms as much as possible. Calculate and submit the time and space complexities for each of the algorithms you propose.

D. Please do not copy from the internet or any other source. Suitable plagiarism checkers would be applied and if found to be copied, the corresponding submission would receive zero marks.

E. Each submission should be a single zip/tar.gz file containing all the codes and other necessary files. From each group only one person should submit. The group number should be mentioned in the submitted file name.

1. [STACK] The n-queens puzzle is the problem of placing n queens on an n x n chessboard such that no two queens attack each other. Standard chess rules apply. A queen can attack any piece horizontally, vertically as well as diagonally anywhere on the board. Given an integer n, return a solution to the n-queens puzzle.

2. [STACK] A knight's tour is a sequence of moves of a knight on a chessboard such that the knight visits every square exactly once. Standard chess rules apply. A knight can move two squares in one direction and one square either horizontally or vertically, in either order. Given an integer n, find a knight's tour of an n x n chessboard.

3. [QUEUE] Given a stream of characters, print the first non-repeating character for each new character arriving in the stream. At any point during the input stream, if there is a repeating character, print '-' as output. You can assume that '-' will not be present as any of the input characters.

Input:	a a b c	Input:	a a c
Output:	a - b b	Output:	a - c

4. [QUEUE] Given that integers are read from a data stream. Find the median of elements read so far in an efficient way.

Input:	5 15 1 3
Output:	5 10 5 4

At any point, if the number of inputs seen till now are even, the median is the average of two middle elements in the sorted stream.

[Hint: Use priority queues]

5. [LINKED LIST] Write a program to merge two unsorted linked lists. Given lists l1 = (4, 2, 7) and l2 = (5, 1), after return from merge the list should be changed to sorted list = (1,2,4,5,7).

   Input: List 1 = 3 -> 1 -> 5, List 2 = 6-> 2 -> 4
   Output: 1 -> 2 -> 3 -> 4 -> 5 -> 6

   Input: List 1 = 4 -> 7 -> 5, List 2 = 2-> 1 -> 8 -> 1
   Output: 1 -> 1 -> 2 -> 4 -> 5 -> 7 -> 8

6. [LINKED LIST] Write a program to print the median value in a sorted linked list. If the length i of the list is odd, then the median is the ceiling(i/2) member. For example, given the list (1, 2, 2, 5, 7, 9, 11) as input, your function should return the value 5. If the length of the list is even, then the median is the mean of the i/2 and (i/2)+1 members. Thus, the median of the sorted list (2, 4, 8, 9) is (4+8)/2. Finally, define the median of an empty list to be 0

7. [SEARCHING/SORTING] In an input sequence, for example (20, 30, 4, 5, 1, 42, 36, 100, 200). First arrange the numbers in a separate list according to the number of digits (for example 20, 30, 42, 36 will be in the same group) then sort each list separately. Use the suitable data structure and sorting algorithm. Finally display the completely sorted array.

8. [SEARCHING/SORTING] For a given input sequence, suppose there is a limited space. For example, for input sequence = 20, 83, 72, 42, 33, 57, say, the available memory space is 10. In such a situation each number n in the sequence will be stored in location n%10. Eg. 20 will be stored in 20%10 = 0 (index), 33 will be stored in 33%10 = 3 (index). Clearly, more than one number may fall into the same index (for example 72 and 42 will fall in the same index 2), in that case store the numbers in an additional data structure in the order they appear in the input sequence (for example 72, 42). Elements who fall in the same index have to be sorted separately (for example 83, 33 have to be sorted, 72, 42 have to be sorted and so on). Apply suitable data structure and sorting algorithm and display the final sorted list.