Assignment 01

Topics: 1D, 2D Arrays, Elementary Sorting, and Linked List

Note: You have to submit a .cpp file for every question. The examples of input and output are for understanding only. Your code should be generalized i.e., work on any related test case.
The deadline for the assignment is a day before the mid-1 exam. No late submissions will be entertained.
In case of plagiarism, straight zero marks will be assigned.

1D and 2D Arrays	Marks: 20

Q1: You are given an array of size N containing distinct integers. An inversion in an array occurs when there are two indices i and j such that i < j but arr[i] > arr[j]. Your task is to implement a C++/JAVA program to count the number of inversions in the array efficiently, where a positive integer N represents the size of the array of N distinct integers and is entered as input by the user.
Example: Input: N = 6
Array = [5, 3, 9, 1, 7, 6]
Output:
(5, 3) is an inversion because 5 > 3.
(5, 1) is an inversion because 5 > 1.
(9, 1) is an inversion because 9 > 1.
(7, 6) is an inversion because 7 > 6.
(9, 7) is an inversion because 9 > 7.
(9, 6) is an inversion because 9 > 6.
(3, 1) is an inversion because 3 > 1.
So, there are a total of 7 inversions in the given array.

Q2:
Develop a program to manage seating arrangements for a conference hall. The hall has n rows and m columns of seats. Each seat can either be occupied or vacant. Your task is to write a C++/JAVA program that initializes a 2D array representing the seating arrangement, where each element represents a seat (1 for occupied, 0 for vacant). Then, implement a function to search for consecutive three vacant seats either in the same row or the same column. Your program should through an error in case of any wrong index.

- Prompt the user to enter the size of the seating grid (n rows and m columns).
- Initialize 2D array representing the seating arrangement with randomly assigned occupied(1) or vacant (0) seats.
- Implement a function to search for consecutive three vacant seats either in the same row or the same column.
- Display all the coordinates (rows and columns) which satisfy the condition.

Sorting	Marks: 30

Q3: You have some items on your online platform. Your platform allows customers to filter items based on both the age group they are suitable for and the price. You must create an object array. To provide a better user experience, you want to display the items in the following order:

- First, sort items by age in ascending order, so that items suitable for younger customers appear first.
- Then, within each age group, sort items by price in descending order, so that the most expensive items come first within each age group.

Write the code (C++/Java) to sort the object array according to the given requirements.
E.g. Before Sorting:

Age	4	8	6	4	8	2	6	2	10	4
Price	15.99	9.99	19.99	7.49	12.99	5.99	14.49	8.99	6.99	10.99
Item	Shirt	Pants	Pants	Pants	Shirt	Shirt	Pants	Pants	Pants	Shirt

After Sorting:

Age	2	2	4	4	4	6	6	8	8	10
Price	8.99	5.99	15.99	10.99	7.49	19.99	14.49	12.99	9.99	6.99
Item	Shirt	Pants	Shirt	Shirt	Pants	Pants	Pants	Shirt	Pants	Pants

Q4: Can we make the selection sort algorithm adaptive? If yes, write a C++/JAVA program for adaptive selection sort.

Q5: Explore the bidirectional bubble sort. Write its C++/JAVA code. Explain if it is better in terms of sorting properties (stable, adaptive, and in place) than the adaptive bubble sort.

Linked List	Marks: 20

Q6: You are given an Abstract Data Type (ADT) of a singly linked list in which **“head”** is preserved. Each node of the list holds an integer value. Your task is to implement a function that takes an integer **“n”** as an argument and splits the linked list into **“k”** sublists in such a way that the sum of the data in each sublist is equal to **“n”**.

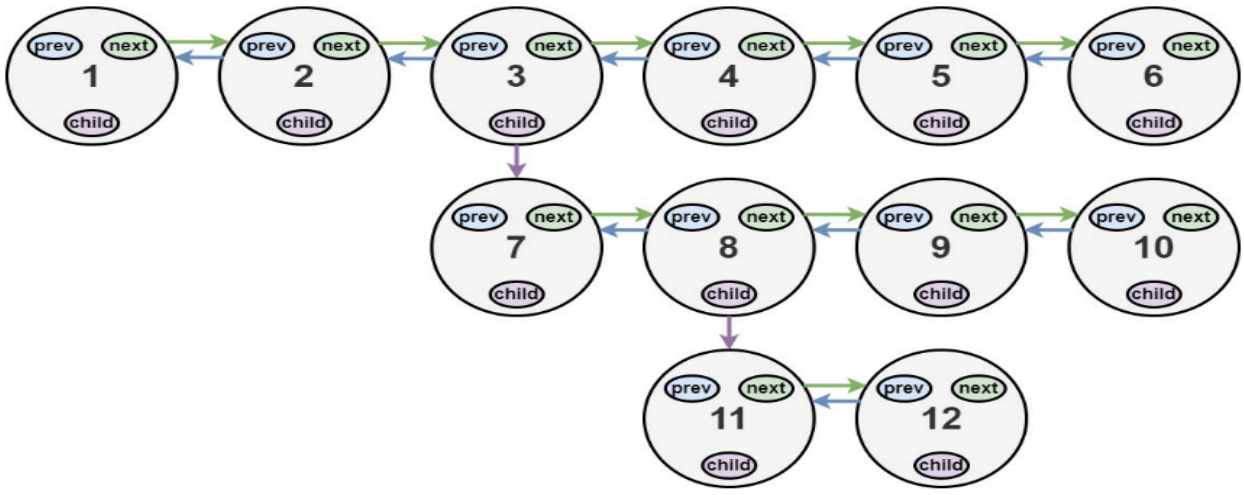
Test Case 1:	**Test Case 2:**
Original List: *10->5->4->11->3->2->1* N = 16	**Original List:** *1->5->4->7->3->2->6* N = 11
Sublist_1: *11->3->2* **Original List:** *10->5->4->1*	**Sublist_1:** *4->7* **Sublist_2:** *3->2->6* **Original List:** *1->5*

Constraints:

- You can assume that the linked list contains at least **"k"** nodes.
- Range for **"k"** is from **"0 to n"**.
- The sum of data in each sublist should be exactly equal to n.
- Each sublist need only contain consecutive nodes from the original list.
- A sublist, that satisfies the given criteria, must be removed from the original list.
- You need to create as many sublists as possible.
- The original order of nodes in the linked list should be preserved during the splitting process.
- The nodes which don't satisfy the condition will be left in the original linked list.

Q7: You are given a doubly linked list, which contains nodes that have a next pointer, a previous pointer, and an additional **child pointer**. This child pointer may or may not point to a separate doubly linked list, also containing these special nodes. These child lists may have one or more children of their own, and so on, to produce a **multilevel data structure** as shown in the example below.
Given the head of the first level of the list, flatten the list so that all the nodes appear in a single-level, doubly linked list. Let curr be a node with a child list. The nodes in the child list should appear after curr and before curr->next in the flattened list.
Return the head of the flattened list. The nodes in the list must have all of their child pointers set to null.

Example 1:



Input: head = [1,2,3,4,5,6,null,null,null,7,8,9,10,null,null,11,12]
Output: [1,2,3,7,8,11,12,9,10,4,5,6]
Explanation: The multilevel linked list in the input is shown.
After flattening the multilevel linked list it becomes: