Fast**National University of Computer & Emerging Sciences, Karachi  
Fall 2018 CS-Department  
Mid Examination-001  
23rd Oct 2018, 8:30 am – 10:30 am**

|  |  |  |
| --- | --- | --- |
| **Course Code: CL 201** | **Course Name: Data Structures Lab** | |
| **Instructor Name / Names: SafiaBaloch, Faizan Yousuf, MahamMobin** | | |
| **Student Roll No:** | | **Section:** |

**Instructions:**

* Return the question paper.
* Read question completely before answering it.
* In case of any ambiguity, you may make assumption. But your assumption should not contradict any statement in the question paper.
* All the answers must be solved according to the sequence given in the question paper.

**Time: 120 minutes.** **Max Marks: 40**

**Recursion: (20)**

**Problem 1:**Given an integer n. Print first n elements of [Recaman’s sequence](http://mathworld.wolfram.com/RecamansSequence.html).

Examples:

Input : n = 6

Output : 0, 1, 3, 6, 2, 7

Input : n = 17

Output : 0, 1, 3, 6, 2, 7, 13, 20, 12, 21,

11, 22, 10, 23, 9, 24, 8

It is basically a function with domain and co-domain as natural numbers and 0. It is recursively defined as below:  
Specifically, let a(n) denote the (n+1)-th term. (0 being already there).  
The rule says:

a(0) = 0,

if n > 0 and the number is not

already included in the sequence,

a(n) = a(n - 1) - n

else

a(n) = a(n-1) + n.

**Problem 2:**The Alexander Bogomolyn’s algorithm is used to permute first N natural numbers.  
Given the value of N we have to output all the permutations of numbers from 1 to N.

**Examples:**

Input : 2

Output : 1 2

2 1

Input : 3

Output : 1 2 3

1 3 2

2 1 3

3 1 2

2 3 1

3 2 1

The idea is to maintain an array to store the current permutation. A static integer level variable is used to define these permutations.

1. It initializes the value of current level and permutes the remaining values to the higher levels.
2. As the assigning action of the values reaches to the highest level, it prints the permutation obtained.
3. This approach is recursively implemented to obtain all possible permutations.

**Sorting: (10)**

Bangalore City, where peace prevails *most* of the time. Not everyone is a huge fan of peace, though. Certainly not **Mr. XYZ**, whose identity is not known to us - yet. Mr. XYZ has somehow managed to bring vampires and zombies to Bangalore City to attack and destroy the city.

[*Fatal Eagle*](https://www.hackerearth.com/users/fataleagle/)*,* an ordinary citizen of the city is extremely worried on seeing his city being attacked by these weird creatures. But, as of now, he has no power to stop these creatures from their silent attacks. He wants to analyze these creatures firstly. He figured out some things about these creatures, like:

1. Zombies have power in terms of an EVEN number.
2. Vampires have power in terms of an ODD number.

If he sees a zombie or a vampire, he marks them in his list with their power. After generating the entire list of power of these creatures, he decides to arrange this data in the following manner:

* All the zombies arranged in sorted manner of their power, followed by the total power of zombies.
* All the vampires arranged in sorted manner of their power, followed by the total power of vampires.

You've to help him produce the following list to help him save his city.

**Input constraints:**  
The first line of input will contain an integer — **N**, denoting the number of creatures. The next line will contain **N** integers denoting the elements of the list containing the power of zombies and vampires.

**Output constraints:**  
Print the required list in a single line.

**Constraints:**  
1 ≤ **N** ≤ 103  
1 ≤ **Ni** ≤ 103

SAMPLE INPUT

6

2 3 10 12 15 22

SAMPLE OUTPUT

2 10 12 22 46 3 15 18

**Linked List: (10)**

Create a Double Link List and implement the following function:

altNode() prints alternate nodes of the Linked List, first from head to end, and then from end to head. If Linked List has even number of nodes, then altNode() skips the last node.

For Linked List 1->2->3->4->5, altNode() prints 1 3 5 5 3 1.

For Linked List 1->2->3->4->5->6, altNode() prints 1 3 5 5 3 1.