Problem A

In this problem, you will have a linked list of sorted linked lists containing integers. You will have to merge the linked lists of integers into a single sorted linked list of integers.

You must use the given template.

Input:
First line: n, the number of linked lists. (1<=n<=100)
For 1<=i<=n:

Next line: m_i, an integer (1<=v <=10000), the number of integers in i-th linked list.
Next m_i lines: v_j, an integer (1<=j<=m_i, -2147483648<=v_j <=2147483647), the j-th value of the i-th linked list.

Output:
Each line contains the values in the sorted linked list.

Sample Case:

Input	Output
3 3 3 5 9 2 4 7 2 6 8	3 4 5 6 7 8 9

Problem B

In this problem, you will have a ternary tree of integers where each node has up to three children: left child, mid-child, and right child. You will have to print the tree in a new order that prints the tree in the following order: left sub-tree, mid sub-tree, node value, and finally the right sub-tree.

Input:
First line: r, the value of root. (-2147483648<=r<=2147483647)
Next line: n, the number of operations. (1<=n<=10000)
Next n lines: *op key val*, three integers (0<=*op*<=2, -2147483648<=*key, val*<=2147483647). If op = 0, set the left child of the node with the value *key* to *val*. If op = 1, set the mid child of the node with the value *key* to *val*. If op = 2, set the right child of the node with the value *key* to *val*. If the key is not found, ignore the command.

Output:
Each line contains the values in the tree according to the new order.

Sample Case:

Input	Output
5 5 0 5 6 1 5 7 2 5 8 0 7 9 2 7 10	6 9 7 10 5 8

Problem C

In this problem, you will have to check whether two binary trees are equal or not.

Input:
First line: r1, the value of root of the first tree. (-2147483648<=r1<=2147483647)
Next line: n1, the number of operations. (1<=n1<=10000)
Next n lines: *op key val*, three integers (0<=*op*<=1, -2147483648 <= *key, val* <= 2147483647). If op = 0, set the left child of the node in the first tree with the value *key* to *val*. If op = 1, set the right child of the node in the first tree with the value *key* to *val*. If the key is not found, ignore the command.
Next line: r2, the value of root of the second tree. (-2147483648<=r2<=2147483647)
Next line: n2, the number of operations. (1<=n2<=10000)
Next n lines: *op key val*, three integers (0<=*op*<=1, -2147483648 <= *key, val* <= 2147483647). If op = 0, set the left child of the node in the second tree with the value *key* to *val*. If op = 1, set the right child of the node in the second tree with the value *key* to *val*. If the key is not found, ignore the command.

Output:
1, if the trees are equal. 0, otherwise.

Sample Case:

Input	Output
5 4 0 5 6 1 5 7 0 7 9 1 7 10 5 4 0 5 6 1 5 9 1 9 10 0 10 7	0
5 4 0 5 6 1 5 7	1

0 7 9 1 7 10 5 4 0 5 6 1 5 7 1 7 10 0 7 9	

Problem D

Input:
First line: n, a number (1<=n<=15).
Second line: k, a number (1<=k<=n).

Output:
Print all possible combinations of k digits from (0-n) in descending order.

Sample Case:

Input	Output
5 3	4 3 2 4 3 1 4 3 0 4 2 1 4 2 0 4 1 0 3 2 1 3 2 0 3 1 0 2 1 0

Problem E

Input:
First line: n, a number (1<=n<=10).

Output:
Each line will contain a list of space-separated digits where,

- The list contains n digits.
- The digits in the list are 0-4 (inclusive)
- The odd indexed digit in the list is an even digit
- The even indexed digit in the list is an odd digit

The lists will be printed in descending order. Example: {1 0 3} before {1 0 1}.

Sample Case:

Input	Output
2	3 4 3 2 3 0 1 4 1 2 1 0
3	3 4 3 3 4 1 3 2 3 3 2 1 3 0 3 3 0 1 1 4 3 1 4 1 1 2 3 1 2 1 1 0 3 1 0 1