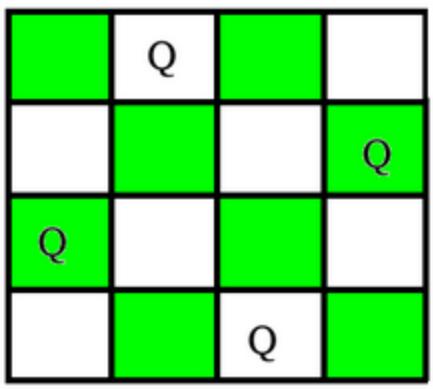
KARPAGAM COLLEGE OF ENGINEERING

17PE05/17FE05/17ME33/17LE33/17TE33/17EE33/17NE33
DESIGN AND ANALYSIS OF ALGORITHMS

SESSION 3.3

1.	Given a sorted array keys[0.. n-1] of search keys and an array freq[0.. n-1] of frequency counts, where freq[i] is the number of searches to keys[i]. Construct a binary search tree of all keys such that the total cost of all the searches is as small as possible. Let us first define the cost of a BST. The cost of a BST node is level of that node multiplied by its frequency. Level of root is 1. **Input:** First line consists of test cases T. First line of every test case consists of N, denoting the number of key. Second and Third line consists N spaced elements of keys and frequency respectively. **Output:** Print the most minimum optimal cost. **Constraints:** 1<=T<=100 1<=N<=100 **Example:** **Input:** 2 2 10 12 34 50 3 10 12 20 34 8 50 **Output:** 118 142
2.	The n-queens puzzle is the problem of placing n queens on an n×n chessboard such that no two queens attack each other. Given an integer n, print all distinct solutions to the n-queens puzzle. Each solution contains distinct board configurations of the n-queens' placement, where the solutions are a permutation of [1,2,3..n] in increasing order, here the number in the *ith* place denotes that the *ith*-column queen is placed in the row with that number. For eg below figure represents a chessboard [3 1 4 2]. 

	Input: The first line of input contains an integer **T** denoting the no of test cases. Then T test cases follow. Each test case contains an integer n denoting the size of the chessboard. **Output:** For each test case, output your solutions on one line where each solution is enclosed in square brackets '[', ']' separated by a space . The solutions are permutations of {1, 2, 3 …, *n*} in increasing order where the number in the ith place denotes the ith-column queen is placed in the row with that number, if no solution exists print -1. **Constraints:** 1 <= T <= 10 1 <= n <= 10 **Example:** **Input** 2 1 4 **Output:** [1] [2 4 1 3] [3 1 4 2]
3.	Given an array of integers and a sum, the task is to print all subsets of given array with sum equal to given sum. Examples: Input : arr[] = {2, 3, 5, 6, 8, 10} sum = 10 Output : 5 2 3 2 8 10 Input : arr[] = {1, 2, 3, 4, 5} sum = 10 Output : 4 3 2 1 5 3 2 5 4 1