Question no 1

**Problem Statement**

The Tower of Hanoi is a famous puzzle where we have three rods and N disks. The objective of the puzzle is to move the entire stack to another rod. You are given the number of disks N. Initially, these disks are in rod 1. You need to print all the steps of disk movement so that all the discs reach the 3rd rod. Also, you need to find the total moves.

**Note:** The disks are arranged such that the top disk is numbered 1 and the bottom-most disk is numbered N. Also, all the disks have different sizes, and a bigger disk cannot be put on top of a smaller disk.

**Input format :**

The input consists of a single integer **N**, representing the number of disks initially placed on rod 1.

**Output format :**

The output prints a series of lines representing each move of a disk from one rod to another.

After all moves are printed, the last line prints the total number of moves made during the Tower of Hanoi puzzle.

**Refer to the sample output for formatting specifications.**

**Code constraints :**

1 ≤ N ≤ 8

**Sample test cases :**

**Input 1 :**

2

**Output 1 :**

move disk 1 from rod 1 to rod 2

move disk 2 from rod 1 to rod 3

move disk 1 from rod 2 to rod 3

3

**Input 2 :**

3

**Output 2 :**

move disk 1 from rod 1 to rod 3

move disk 2 from rod 1 to rod 2

move disk 1 from rod 3 to rod 2

move disk 3 from rod 1 to rod 3

move disk 1 from rod 2 to rod 1

move disk 2 from rod 2 to rod 3

move disk 1 from rod 1 to rod 3

7

Question no 2

**Problem Statement**

John is organizing data in a binary search tree for quick retrieval. He needs a program that builds the tree and recursively performs inorder, preorder, and postorder traversals to display the elements in different orders.

Help him implement the solution by creating the tree and performing the required traversals.

**Input format :**

The input consists of integers representing the choice and the corresponding operation.

If the choice is 1, enter the number of elements (n) and the elements to be inserted into the tree.

If the choice is 2, print the in-order traversal.

If the choice is 3, print the pre-order traversal.

If the choice is 4, print the post-order traversal.

If the choice is 5, exit.

**Output format :**

The output prints the results based on the choice.

**Refer to the sample output for formatting specifications.**

**Code constraints :**

The given test cases fall under the following constraints:

1 ≤ n ≤ 15

1 ≤ BST elements ≤ 1000

**Sample test cases :**

**Input 1 :**

1

5

12 78 96 34 59

2

3

4

5

**Output 1 :**

Inorder: 12 34 59 78 96

Preorder: 12 78 34 59 96

Postorder: 59 34 96 78 12

**Input 2 :**

1

9

7 9 6 3 2 1 4 5 8

2

4

5

**Output 2 :**

Inorder: 1 2 3 4 5 6 7 8 9

Postorder: 1 2 5 4 3 6 8 9 7

Question no 3

**Problem Statement**

Kamal is interested in managing a collection of integers using a Binary Search Tree (BST). He wants to write a program that constructs a BST from a given set of integers and then deletes all the nodes with values greater than the average value of all nodes in the BST.

Help him solve this problem.

**Example**

**Input:**

5

10 20 30 40 50

**Output:**

10 20 30

**Explanation:**

The sum of values is 10 + 20 + 30 + 40 + 50 = 150, and there are 5 nodes. So, the average is 150 / 5 = 30. Then, delete nodes with values greater than the average (30). The final output is the values of the remaining nodes in the BST, which are 10, 20 and 30.

**Input format :**

The first line consists of an integer **N,** representing the number of elements in the BST.

The second line consists of **N** space-separated integers, representing theBST elements.

**Output format :**

The output prints the remaining integers in the BST after deleting nodes greater than the average value, separated by a space.

The integers should be printed in an in-order traversal.

**Refer to the sample output for the formatting specifications.**

**Code constraints :**

The given test cases will fall under the following constraints:

1 ≤ N ≤ 20

1 ≤ elements ≤ 1000

**Sample test cases :**

**Input 1 :**

5

10 20 30 40 50

**Output 1 :**

10 20 30

**Input 2 :**

6

20 52 65 98 74 57

**Output 2 :**

20 52 57

Question no 4:

**Problem Statement**

Sam needs to sort an array of integers using the divide-and-conquer method. He wants to implement the merge sort algorithm, displaying the array after each iteration to track the sorting process.

Assist him in writing a program that divides the array, merges it back, and prints the array.

**Input format :**

The first line of input consists of an integer**n,** denoting the array size.

The second line consists of **n** space-separated integers, representing the array of elements.

**Output format :**

The first line of output prints the given array.

The following lines print the sorted array, after each iteration.

The last line of output prints the final sorted array.

**Refer to the sample output for formatting specifications.**

**Code constraints :**

The given test cases fall under the following constraints:

1 ≤ n ≤ 20

0 ≤ array elements ≤ 50

**Sample test cases :**

**Input 1 :**

6

4 1 5 3 2 6

**Output 1 :**

Given Array

4 1 5 3 2 6

After iteration 1

1 4 5 3 2 6

After iteration 2

1 4 5 3 2 6

After iteration 3

1 4 5 2 3 6

After iteration 4

1 4 5 2 3 6

After iteration 5

1 2 3 4 5 6

Sorted Array

1 2 3 4 5 6

**Input 2 :**

8

8 1 7 2 6 3 5 4

**Output 2 :**

Given Array

8 1 7 2 6 3 5 4

After iteration 1

1 8 7 2 6 3 5 4

After iteration 2

1 8 2 7 6 3 5 4

After iteration 3

1 2 7 8 6 3 5 4

After iteration 4

1 2 7 8 3 6 5 4

After iteration 5

1 2 7 8 3 6 4 5

After iteration 6

1 2 7 8 3 4 5 6

After iteration 7

1 2 3 4 5 6 7 8

Sorted Array

1 2 3 4 5 6 7 8

question no 5

**Problem Statement**

Ravi is spending quality time with his children, teaching them about ascending order through a fun game. To make it more interactive, Ravi has decided to create a simple program that takes a list of characters and demonstrates how to arrange them in ascending order.

Guide Ravi by implementing this program using quick sort.

**Input format :**

The first line of input consists of an integer **n,** representing the number of characters.

The next line consists of **n** space-separated characters (lowercase letters or uppercase letters).

**Output format :**

The output displays a single line containing the n characters sorted in ascending order with ASCII values, separated by spaces.

**Refer to the sample output for formatting specifications.**

**Code constraints :**

1 ≤ n ≤ 26

**Sample test cases :**

**Input 1 :**

5

s A a n h

**Output 1 :**

A a h n s

**Input 2 :**

6

h g f d v a

**Output 2 :**

a d f g h v

**Input 3 :**

5

h a p p y

**Output 3 :**

a h p p y