**Problem Statement 1**

Shyam is organizing a sports event where athletes are ranked based on their scores. He needs to ensure that the highest-ranked athlete is always at the top of the list, so he arranges the scores in a **max heap**. However, due to an injury, the top-ranked athlete has to drop out of the competition.

Shyam needs to remove the athlete and maintain the ranking system for the remaining athletes. Help Shyam by constructing the max heap and removing the root (highest score) while keeping the heap property intact.

**Input format :**

The first line contains an integer n, representing the number of athletes.

The second line contains n space-separated integers representing the scores of the athletes.

**Output format :**

The output prints the scores of the athletes after building the max heap and removing the highest score (root of the heap).

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

**Code constraints :**

The test cases will be under the following constraints:

1 ≤ n ≤ 10

1 ≤ score ≤ 100

The input array should not have duplicate elements.

**Sample test cases :**

**Input 1 :**

5

23 25 2 12 52

**Output 1 :**

25 23 2 12

**Input 2 :**

5

10 5 3 2 4

**Output 2 :**

5 4 3 2

**Problem Statement 2**

Mani manages a collection of fragile items that must be carefully arranged so the lightest ones are always prioritized for delivery, forming a **min-heap**. After sorting the items, Mani realizes one specific item is damaged and needs to be removed from the collection.

Help Mani by first arranging the items based on their weight using a min-heap, and then removing the damaged item while ensuring the collection remains properly organized according to the min-heap property.

**Input format :**

The first line contains an integer n, representing the number of items Mani needs to organize.

The second line contains n space-separated integers representing the weights of the items.

The third line contains an integer key, the weight of the item Mani needs to remove.

**Output format :**

The output prints the weights of the items after building the min heap and removing the specified item.

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

**Code constraints :**

The test cases will fall under the following constraints:

1 ≤ N ≤ 10

1 ≤ item\_weight ≤ 100

1 ≤ key ≤ 100

The key will always be one of the elements in the input.

**Sample test cases :**

**Input 1 :**

5

2 5 10 29 49

10

**Output 1 :**

2 5 49 29

**Input 2 :**

6

14 25 36 95 75 84

75

**Output 2 :**

14 25 36 95 84

**Problem Statement 3**

Emma is working with a collection of numbers and needs to manage them using a min-heap data structure. She wants to insert a set of numbers into the **min-heap**, and then she needs to delete all elements within a specified range (inclusive).

Your task is to help Emma by implementing the necessary operations to build the min-heap, display its contents, and remove elements in the specified range.

You need to:

* Insert numbers into the min-heap.
* Delete all elements within the specified range.
* Display the contents of the min-heap before and after deletion.

**Input format :**

The first line consists of an integer **n,** the number of elements in the array.

The second line of input consists of**n**elements separated by spaces.

The third line consists of the **start** range and **end** range, separated by a space.

**Output format :**

The output displays the following format:

First, print the contents of the min-heap after all elements have been inserted.

Second, print the contents of the min-heap after deleting the specified range of elements.

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

**Code constraints :**

The test cases will fall under the following constraints:

1 ≤ n ≤ 10

1 ≤ elements ≤ 100

1 ≤ start ≤ end ≤ 100

range should be given input elements.

**Sample test cases :**

**Input 1 :**

7

20 36 82 17 51 3 96

20 96

**Output 1 :**

Min-heap: 3 20 17 36 51 82 96

Min-heap after deletion: 3 17

**Input 2 :**

4

50 40 30 20

10 25

**Output 2 :**

Min-heap: 20 30 40 50

Min-heap after deletion: 30 50 40

**Problem Statement 4**

Tao is working on a data processing application that analyzes a list of integers. He wants to implement a feature that processes the data in the following manner:

* Build a min-heap from a given list of integers.
* Calculate the average of the integers in the list.
* Filter out the integers that are less than the calculated average and maintain the min-heap property for the remaining integers.

**Input format :**

The first line contains an integer n, representing the number of integers in the list.

The second line contains n space-separated integers, representing the values.

**Output format :**

The first line displays "Average: " representing a double representing the average of the integers, rounded to two decimal places.

The second line displays the remaining integers in the modified min-heap, space-separated.

**Refer to the sample output for the exact format.**

**Code constraints :**

the test cases will fall under the following constraints:

1 ≤ n ≤ 10

1 ≤ values ≤ 100

**Sample test cases :**

**Input 1 :**

5

12 76 17 34 72

**Output 1 :**

Average: 42.20

72 76

**Input 2 :**

8

83 53 45 51 59 54 30 78

**Output 2 :**

Average: 56.62

59 83 78

**Problem Statement 5**

Ram is working with a **Max Heap** to manage a set of numbers. He needs your help to build the heap, delete the root element, and calculate the sum of all remaining leaf nodes in the heap after the deletion. Your task is to:

* Insert numbers into a Max Heap.
* Print the Max Heap before and after deleting the root.
* Calculate and print the sum of all the leaf nodes in the heap after the root deletion.

Help Ram perform these operations efficiently.

**Input format :**

The first line contains an integer n, representing the number of elements Ram will insert into the heap.

The second line consists of n space-separated integers, representing the elements to be inserted into the heap.

**Output format :**

The output displays the following format:

First, print the Max Heap before the root deletion on one line.

Next, print the Max Heap after the root deletion on one line.

Finally, print the sum of all leaf nodes in the format: "Sum of leaf nodes: X", where X is the computed sum.

**Refer to the sample output for the exact format.**

**Code constraints :**

the test cases will fall under the following constraints:

1 ≤ n ≤ 10

1 ≤ values ≤ 100

**Sample test cases :**

**Input 1 :**

5

4 3 2 5 6

**Output 1 :**

6 5 2 3 4

5 4 2 3

Sum of leaf nodes: 5

**Input 2 :**

6

4 5 6 3 2 1

**Output 2 :**

6 4 5 3 2 1

5 4 1 3 2

Sum of leaf nodes: 6

**Problem Statement 6**

David is a data analyst who often needs to analyze numbers by extracting their digits and sorting them in descending order. This will help him visualize the frequency of each digit and understand patterns within large numerical datasets.

Write a program that takes a number as input, extracts its digits, sorts them, and displays the sorted digits as a single number using the **heap sort algorithm**.

**Input format :**

The input consists of an integer n, representing a single number.

**Output format :**

The output displays the digits of the number sorted in descending order as a single integer without any spaces.

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

**Code constraints :**

The test cases will fall under the following constraints:

2 ≤ n ≤ 109

**Sample test cases :**

**Input 1 :**

123

**Output 1 :**

321

**Input 2 :**

31524

**Output 2 :**

54321

**Input 3 :**

1001

**Output 3 :**

1100

**Problem Statement 7**

Tina is a passionate writer who has recently decided to organize her collection of characters from various stories she has written.

To efficiently manage her collection, she wants to sort the characters' names in alphabetical order using the **heap sort algorithm**.

**Input format :**

The first line contains an integer n representing the number of character names Tina has.

The second line consists of the characters, representing a character's name.

**Output format :**

The output displays the sorted character names in alphabetical order, separated by a space.

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

**Code constraints :**

The test cases will fall under the following constraints:

The names should not exceed 50 lowercase characters.

**Sample test cases :**

**Input 1 :**

5

e d a c b

**Output 1 :**

a b c d e

**Input 2 :**

4

z y x w

**Output 2 :**

w x y z

**Problem Statement 8**

Diego runs a candy shop, and he wants to arrange the candy boxes in non-decreasing order of their sweetness levels. He has a list of candies, where each candy's sweetness level is represented as an integer. Diego needs to sort these sweetness levels to display the candies from the least sweet to the most sweet.

You need to help Diego by implementing a heap sort algorithm that will sort the list of candy sweetness levels in ascending order.

**Input format :**

The first line contains an integer n, representing the number of candy boxes.

The second line contains n space-separated integers representing the sweetness levels of the candies.

**Output format :**

The output displays a single line of n space-separated integers representing the sorted sweetness levels in ascending order.

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

**Code constraints :**

The test cases will fall under the following constraints:

1 ≤ n ≤ 10

1 ≤ sweetness level of candies ≤ 1000

**Sample test cases :**

**Input 1 :**

5

275 635 987 145 325

**Output 1 :**

145 275 325 635 987

**Input 2 :**

8

44 52 63 95 98 74 36 65

**Output 2 :**

36 44 52 63 65 74 95 98

**Problem Statement 9**

Tao is a data analyst who has recently been tasked with organizing a list of survey responses. The responses include both even and odd integers representing different scores given by participants.

To facilitate analysis, he wants to sort the scores into two separate groups: one for odd scores and another for even scores, each sorted in ascending order.

Write a program that takes a list of integer scores as input, separates them into odd and even numbers, sorts each group using the heap sort algorithm, and then outputs the sorted lists.

**Input format :**

The first line contains an integer n, representing the total number of scores.

The second line consists of n space-separated integers score, representing the score given by a participant.

**Output format :**

The output displays the sorted character names in alphabetical order, separated by a space.

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

**Code constraints :**

The test cases will fall under the following constraints:

2 ≤ n ≤ 10

1 ≤ Scores ≤ 1000

**Sample test cases :**

**Input 1 :**

10

9 3 2 1 5 6 7 8 10 4

**Output 1 :**

1 3 5 7 9

2 4 6 8 10

**Input 2 :**

5

6 7 8 3 2

**Output 2 :**

3 7

2 6 8

**Problem Statement 10**

Mythili is currently studying different sorting algorithms in her computer science class. She recently learned about Heap Sort, a popular sorting technique that leverages the properties of max and min heaps. Mythili decided to implement both Max Heap Sort and Min Heap Sort to compare their performance.

She wants to write a program that takes an array of integers as input and provides two sorting options:

* **Max Heap Sort**: It sorts the array in descending order using a max heap.
* **Min Heap Sort**: It sorts the array in ascending order using a min-heap.

**Input format :**

The first line contains an integer, 'n', representing the size of the array.

The second line contains 'n' space-separated integers, 'arr[i]', representing the elements of the array.

The third line contains an integer, 'choice' (1 or 2), representing the user's choice of sorting:

1: Perform max-heap sort.

2: Perform min-heap sort.

**Output format :**

The output displays the following format:

If 'choice' is 1, print the sorted array in ascending order after applying max-heap sort, separated by a space.

If 'choice' is 2, print the sorted array in descending order after applying min-heap sort, separated by a space.

If 'choice' is not 1 or 2, print "Invalid choice."

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

**Code constraints :**

The test cases will fall under the following constraints:

1 <= n <= 10

1 <= arr[i] <= 100

**Sample test cases :**

**Input 1 :**

5

35 89 100 45 91

1

**Output 1 :**

35 45 89 91 100

**Input 2 :**

5

35 89 100 45 91

2

**Output 2 :**

100 91 89 45 35

**Input 3 :**

5

35 89 100 45 91

3

**Output 3 :**

Invalid choice