The problem involves inserting into and maintaining a heap.

To start with, you will be given a set of numbers that you insert, as they come in, into an array. This will be the initial heap (the numbers will be ordered in such a manner that they ensure the "heapness" of the array).

Now, you will be given another set of numbers that you have to insert into the heap. After EACH insertion, you have to ensure the "heapness" of the array, as per the algorithm discussed in the class.

Input

The first line will have a number N, followed by N integers (on the same line, separated by a space). These N integers form the initial heap (they will be already in an order that ensures heapness). The next line will have an integer M, followed by M integers, that have to be entered into the heap, ensuring its heapness after EACH insertion. (M + N <= 50).

Output

On the next line, there will be a number P. You have to print the element at that index, its left child, the left child’s left child, and so on, till you reach the leaf node and print that out too, all on a single line, separated by a space each.

**Input**

6 10 8 5 4 3 2   /\* NOT PART OF INPUT!!! \*/

4 15 11 6 13 /\*after inserting 13, the array should look like: 15 13 10 8 11 2 5 4 6 3 \*/

1

**Output**

13 8 4

Test case

6 10 8 5 4 3 2

4 15 11 6 13

1

Test case

1 10

6 11 9 12 13 7 20

0

#include<stdio.h>

#include<stdlib.h>

int a[100];

int main()

{ int i,j,IN,SN,A;

int n1,root,node;

scanf("%d",&IN);

for (i=0; i < IN; i++)

scanf("%d",a+i);

/\* for (i=0; i < IN; i++)

printf("%d ",a[i]);

printf("\n"); \*/

scanf("%d",&SN);

for (i=0; i < SN; i++)

{

scanf("%d",&n1);

node = i+IN;

a[node] = n1;

while (node > 0)

{

root = (node+1)/2 - 1;

if (a[root] < a[node])

swap(root,node);

else break;

node = root;

}

/\* for (j=0; j <= IN+i; j++)

printf("%d ",a[j]); \*/

}

scanf ("%d",&A);

while (A < IN+SN) {

printf("%d ",a[A]);

A = (A+1)\*2 - 1;

}

printf("\n");

return 0;

}

swap(r,b)

int r,b;

{

int t = a[r];

a[r] = a[b]; a[b] = t;

}