Objective

In this challenge, we're going to use loops to help us do some simple math. Check out the Tutorial tab to learn more.

Task

Given an integer, ***n***, print its first ***10*** multiples. Each multiple ***n X i*** (where ***1 ≤ i ≤ 10***) should be printed on a new line in the form: n x i = result.

Input Format

A single integer, ***n***.

Constraints

2 ≤ n ≤ 20

Output Format

Print ***10*** lines of output; each line ***i*** (where ***1 ≤ i ≤ 10***) contains the ***result*** of ***n X i*** in the form:

n x i = result.

Sample Input

2

Sample Output

2 x 1 = 2

2 x 2 = 4

2 x 3 = 6

2 x 4 = 8

2 x 5 = 10

2 x 6 = 12

2 x 7 = 14

2 x 8 = 16

2 x 9 = 18

2 x 10 = 20

Answer: (penalty regime: 0 %)

Ace editor not ready. Perhaps reload page?
Falling back to raw text area.

```c
#include<stdio.h>
int main(){
    int n;
    scanf("%d",&n);
    for(int i=1;i<=10;i++){
        printf("%d x %d = %d\n",n,i,n*i);
    }
}
```

	Input	Expected	Got	
✓	2	2 x 1 = 2 2 x 2 = 4 2 x 3 = 6 2 x 4 = 8 2 x 5 = 10 2 x 6 = 12 2 x 7 = 14 2 x 8 = 16 2 x 9 = 18 2 x 10 = 20	2 x 1 = 2 2 x 2 = 4 2 x 3 = 6 2 x 4 = 8 2 x 5 = 10 2 x 6 = 12 2 x 7 = 14 2 x 8 = 16 2 x 9 = 18 2 x 10 = 20	✓

Passed all tests! ✓

A nutritionist is labeling all the best power foods in the market. Every food item arranged in a single line, will have a value beginning from 1 and increasing by 1 for each, until all items have a value associated with them. An item's value is the same as the number of macronutrients it has. For example, food item with value 1 has 1 macronutrient, food item with value 2 has 2 macronutrients, and incrementing in this fashion.

The nutritionist has to recommend the best combination to patients, i.e. maximum total of macronutrients. However, the nutritionist must avoid prescribing a particular sum of macronutrients (an 'unhealthy' number), and this sum is known. The nutritionist chooses food items in the increasing order of their value. Compute the highest total of macronutrients that can be prescribed to a patient, without the sum matching the given 'unhealthy' number.

Here's an illustration:

Given *4* food items (hence value: *1,2,3* and *4*), and the unhealthy sum being *6* macronutrients, on choosing items *1, 2, 3* -> the sum is *6*, which matches the 'unhealthy' sum. Hence, one of the three needs to be skipped. Thus, the best combination is from among:

- *2 + 3 + 4 = 9*
- *1 + 3 + 4 = 8*
- *1 + 2 + 4 = 7*

Since *2 + 3 + 4 = 9*, allows for maximum number of macronutrients, 9 is the right answer.

Complete the code in the editor below. It must return an integer that represents the maximum total of macronutrients, modulo *1000000007* ($10^9 + 7$).

It has the following:

n: an integer that denotes the number of food items

k: an integer that denotes the unhealthy number

Constraints

- $1 \le n \le 2 \times 10^9$
- $1 \le k \le 4 \times 10^{15}$

Input Format For Custom Testing

The first line contains an integer, *n*, that denotes the number of food items.

The second line contains an integer, *k*, that denotes the unhealthy number.

Sample Input 0

```
2
2
```

Sample Output 0

```
3
```

Explanation 0

The following sequence of *n = 2* food items:

1. Item 1 has 1 macronutrients.
2. *1 + 2 = 3*; observe that this is the max total, and having avoided having exactly *k = 2* macronutrients.

Sample Input 1

```
2
1
```

Sample Output 1

```
2
```

Explanation 1

1. Cannot use item *1* because *k = 1* and *sum ≡ k* has to be avoided at any time.
2. Hence, max total is achieved by *sum = 0 + 2 = 2*.

Sample Case 2

Sample Input For Custom Testing

Sample Input 2

```
3
3
```

Sample Output 2

```
5
```

Explanation 2

2 + 3 = 5, is the best case for maximum nutrients.

Answer: (penalty regime: 0 %)

```c
#include<stdio
int main(){
    int a,n;
    long long
    scanf("%d",
    scanf("%d",&n);
    for(int i=1;i<=a;i++){
        sum+=i;
        if(sum==n){
            sum-=1;
        }
    }
    printf("%lld",sum%1000000007);
}
```

OK

	Input	Expected	Got	
✓	2 2	3	3	✓
✓	2 1	2	2	✓
✓	3 3	5	5	✓

Passed all tests! ✓

Determine all positive integer values that evenly divide into a number, its factors. Return the p^{th} element of your list, sorted ascending. If there is no p^{th} element, return *0*.

For example, given the number $n = 20$, its factors are *{1,2,4,5,10,20}*. Using **1-based indexing** if $p = 3$, return *4*. If $p > 6$, return *0*.

Complete the code in the editor below. The function should return a long integer value of the p^{th} integer factor of *n*.

It has the following:

n: an integer

p: an integer

Constraints

- $1 \le n \le 10^{15}$
- $1 \le p \le 10^{9}$

Input Format for Custom Testing

Input from stdin will be processed as follows and passed to the function.

The first line contains an integer *n*, the number to factor.

The second line contains an integer *p*, the 1-based index of the factor to return.

Sample Input 0

10

3

Sample Output 0

5

Explanation 0

Factoring $n = 10$ we get *{1, 2, 5, 10}*. We then return the $p = 3^{rd}$ factor as our answer.

Sample Input 1

Sample Input 1

10

5

Sample Output 1

0

Explanation 1

Factoring *n = 10* we get *{1, 2, 5, 10}*. There are only *4* factors and *p = 5*. We return *0* as our answer.

Sample Input 2

1

1

Sample Output 2

1

Explanation 2

Factoring *n = 1* we get *{1}*. We then return the $p = 1^{st}$ factor as our answer.

Answer: (penalty regime: 0 %)

```c
#include<stdio.h>
int main(){
    int n;
    scanf("%d",&n);
    int p;
    scanf("%d",&p);
    int arr[n],e=0;
    for(int i=1;i<=n;i++){
        if(n%i==0){
            arr[e]=i;
            e++;
        }
    }
    if(p>e){
        printf("0");
    }else{
        printf("%d",arr[p-1]);
    }
}
```

	Input	Expected	Got	
✓	10 3	5	5	✓
✓	10 5	0	0	✓
✓	1 1	1	1	✓

Passed all tests! ✓

```c
#include<stdio.h>
int main(){
    int t;
    scanf("%d",&t);
    while(t--){
        int n;
        scanf("%d",&n);
        int a[n];
        for(int i=0;i<n;i++){
            scanf("%d",&a[i]);

        }
        int k,flag=0;
        scanf("%d",&k);
        for(int i=0;i<n;i++){
            for(int j=i+1;j<n;j++){
                if((a[i]-a[j]==k||a[j]-a[i]==k)){
                    flag=1;
                    break;
                }
            }
        }
            if(flag==1){

                printf("%d\n",flag);

            }else{
                printf("0");
                printf("\n");
            }
        }
    }

```

	Input	Expected	Got	
✓	1 3 1 3 5 4	1	1	✓
✓	1 3 1 3 5 99	0	0	✓

Passed all tests! ✓

Sam loves chocolates and starts buying them on the 1st day of the year. Each day of the year, x, is numbered from 1 to Y. On days when x is odd, Sam will buy x chocolates; on days when x is even, Sam will not purchase any chocolates.

Complete the code in the editor so that for each day Ni (where $1 \le x \le N \le Y$) in array arr, the number of chocolates Sam purchased (during days 1 through N) is printed on a new line. This is a function-only challenge, so input is handled for you by the locked stub code in the editor.

Input Format

The program takes an array of integers as a parameter.

The locked code in the editor handles reading the following input from stdin, assembling it into an array of integers (arr), and calling calculate(arr).

The first line of input contains an integer, T (the number of test cases). Each line i of the T subsequent lines describes the ith test case as an integer, Ni (the number of days).

Constraints

$1 \le T \le 2 \times 105$
$1 \le N \le 2 \times 106$
$1 \le x \le N \le Y$

Output Format

For each test case, Ti in arr, your calculate method should print the total number of chocolates Sam purchased by day Ni on a new line.

Sample Input 0

```
3
1
2
3
```

Sample Output 0

```
1
1
4
```

Sample Input 0

```
3
1
2
3
```

Sample Output 0

```
1
1
4
```

Explanation

Test Case 0: N = 1
Sam buys 1 chocolate on day 1, giving us a total of 1 chocolate. Thus, we print 1 on a new line.

Test Case 1: N = 2
Sam buys 1 chocolate on day 1 and 0 on day 2. This gives us a total of 1 chocolate. Thus, we print 1 on a new line.

Test Case 2: N = 3
Sam buys 1 chocolate on day 1, 0 on day 2, and 3 on day 3. This gives us a total of 4 chocolates. Thus, we print 4 on a new line.

Answer: (penalty regime: 0 %)

Answer: (penalty regime: 0 %)

```c
#include<stdio.h>
int main(){
    int n,count=0;
    scanf("%d",&n);
    int a[n];
    for(int i=0;i<n;i++){
        scanf("%d",&a[i]);
    }
        for(int j=0;j<n;j++){
            for(int k=1;k<=a[j];k++){
            if(k%2!=0){
                count+=k;
            }
        }
        printf("%d\n",count);
        count=0;
    }
}
```

	Input	Expected	Got	
✓	3 1 2 3	1 1 4	1 1 4	✓
✓	10 71 100 86 54 40 9 77 9 13 98	1296 2500 1849 729 400 25 1521 25 49 2401	1296 2500 1849 729 400 25 1521 25 49 2401	✓

Passed all tests! ✓

The number of goals achieved by two football teams in matches in a league is given in the form of two lists. Consider:

- Football team A, has played three matches, and has scored { 1 , 2 , 3 } goals in each match respectively.
- Football team B, has played two matches, and has scored { 2, 4 } goals in each match respectively.
- Your task is to compute, for each match of team B, the total number of matches of team A, where team A has scored less than or equal to the number of goals scored by team B in that match.
- In the above case:
- For 2 goals scored by team B in its first match, team A has 2 matches with scores 1 and 2.
- For 4 goals scored by team B in its second match, team A has 3 matches with scores 1, 2 and 3.

Hence, the answer: {2, 3}.

Complete the code in the editor below. The program must return an array of m positive integers, one for each maxes[i] representing the total number of elements nums[j] satisfying nums[j] ≤ maxes[i] where 0 ≤ j < n and 0 ≤ i < m, in the given order.

It has the following:

nums[nums[0],...nums[n-1]]: first array of positive integers

maxes[maxes[0],...maxes[n-1]]: second array of positive integers

Constraints

- 2 ≤ n, m ≤ 105
- 1 ≤ nums[j] ≤ 109, where 0 ≤ j < n.
- 1 ≤ maxes[i] ≤ 109, where 0 ≤ i < m.

Input Format For Custom Testing

Input from stdin will be processed as follows and passed to the function.

The first line contains an integer n, the number of elements in nums.
The next n lines each contain an integer describing nums[j] where 0 ≤ j < n.
The next line contains an integer m, the number of elements in maxes.
The next m lines each contain an integer describing maxes[i] where 0 ≤ i < m.

Sample Case 0

Sample Case 0

Sample Input 0

```
4
1
4
2
4
2
3
5
```

Sample Output 0

```
2
4
```

Explanation 0

We are given n = 4, nums = [1, 4, 2, 4], m = 2, and maxes = [3, 5].

1. For maxes[0] = 3, we have 2 elements in nums (nums[0] = 1 and nums[2] = 2) that are ≤ maxes[0].
2. For maxes[1] = 5, we have 4 elements in nums (nums[0] = 1, nums[1] = 4, nums[2] = 2, and nums[3] = 4) that are ≤ maxes[1].

Thus, the function returns the array [2, 4] as the answer.

Sample Case 1

Sample Input 1

```
5
2
10
5
4
```

Sample Case 1

Sample Input 1

```
5
2
10
5
4
8
4
3
1
7
8
```

Sample Output 1

```
1
0
3
4
```

Explanation 1

We are given, n = 5, nums = [2, 10, 5, 4, 8], m = 4, and maxes = [3, 1, 7, 8].

1. For maxes[0] = 3, we have 1 element in nums (nums[0] = 2) that is ≤ maxes[0].
2. For maxes[1] = 1, there are 0 elements in nums that are ≤ maxes[1].
3. For maxes[2] = 7, we have 3 elements in nums (nums[0] = 2, nums[2] = 5, and nums[3] = 4) that are ≤ maxes[2].
4. For maxes[3] = 8, we have 4 elements in nums (nums[0] = 2, nums[2] = 5, nums[3] = 4, and nums[4] = 8) that are ≤ maxes[3].

Thus, the function returns the array [1, 0, 3, 4] as the answer.

Answer: (penalty regime: 0 %)

```c
#include<stdio.h>
int main(){
    int n;
    scanf("%d",&n);
    int a[n],i=0;
    while(i<n){
        scanf("%d",&a[i]);
        i++;
    }
    int m;
    scanf("%d",&m);
    int arr[m],j=0;
    while(j<m){
        scanf("%d",&arr[j]);
        j++;
}
 for(int k=0;k<m;k++){
     int c=0;
     for(int l=0;l<n;l++){
         if(arr[k]>=a[l]){
             c++;
         }
     }
      printf("%d\n",c);
 }
}
```

	Input	Expected	Got	
✓	4 1 4 2 4 2 3 5	2 4	2 4	✓
✓	5 2 10 5 4 8 4 3 1 7 8	1 0 3 4	1 0 3 4	✓

Passed all tests! ✓

Given an array of numbers and a window of size k. Print the maximum of numbers inside the window for each step as the window moves from the beginning of the array.

Input Format

Input contains the array size, no of elements and the window size

Output Format

Print the maximum of numbers

Constraints

1 <= size <= 1000

Sample Input 1

8

1 3 5 2 1 8 6 9

3

Sample Output 1

5 5 5 8 8 9

For example:

Input	Result
8 1 3 5 2 1 8 6 9 3	5 5 5 8 8 9
10 3 7 5 1 2 9 8 5 3 2 3	7 7 5 9 9 9 8 5

Answer: (penalty regime: 0 %)

Answer: (penalty regime: 0 %)

```c
#include<stdio.h>
int main(){
    int n;
    scanf("%d",&n);
    int a[n];
    for(int i=0;i<n;i++){
        scanf("%d ",&a[i]);
    }
    int k;
    scanf("%d",&k);
    int m=k-1;
    for(int i=0;i<n-m;i++){
        int max=a[i];
        for(int j=i+1;j<=i+m;j++){
            if(max<a[j]){
                max=a[j];
            }
        }
        printf("%d ",max);
    }
}
```

	Input	Expected	Got	
✓	8 1 3 5 2 1 8 6 9 3	5 5 5 8 8 9	5 5 5 8 8 9	✓
✓	10 3 7 5 1 2 9 8 5 3 2 3	7 7 5 9 9 9 8 5	7 7 5 9 9 9 8 5	✓

Passed all tests! ✓

Given an array and a threshold value find the output.

Input: {5,8,10,13,6,2}

Threshold = 3

Output count = 17

Explanation:

Number	Parts	Counts
5	{3,2}	2
8	{3,3,2}	3
10	{3,3,3,1}	4
13	{3,3,3,3,1}	5
6	{3,3}	2
2	{2}	1

Input Format

N - no of elements in an array

Array of elements

Threshold value

Output Format

Display the count

Sample Input 1

6

5 8 10 13 6 2

3

Sample Output 1

17

For example:

Input	Result
6 5 8 10 13 6 2 3	17
7 20 35 57 30 56 87 30 10	33

Answer: (penalty regime: 0 %)

```c
#include<stdio.h>
int main(){
    int n;
    scanf("%d",&n);
    int a[n];
    for(int i=0;i<n;i++){
        scanf("%d ",&a[i]);
    }
    int k;
    scanf("%d",&k);
    int sum=0;
    for(int i=0;i<n;i++){
        int count=0;
        int m=a[i]/k;
        count+=m;
        if(a[i]%k!=0){
            count++;
        }
        sum+=count;
    }
    printf("%d",sum);
}
```

	Input	Expected	Got	
✓	6 5 8 10 13 6 2 3	17	17	✓
✓	7 20 35 57 30 56 87 30 10	33	33	✓

Passed all tests! ✓

Output is a merged array without duplicates.

Input Format

N1 - no of elements in array 1

Array elements for array 1

N2 - no of elements in array 2

Array elements for array2

Output Format

Display the merged array

Sample Input 1

5

1 2 3 6 9

4

2 4 5 10

Sample Output 1

1 2 3 4 5 6 9 10

For example:

Input	Result
5 1 2 3 6 9 4 2 4 5 10	1 2 3 4 5 6 9 10

Answer: (penalty regime: 0 %)

Answer: (penalty regime: 0 %)

```c
#include<stdio.h>
int main(){
    int n;
    scanf("%d",&n);
    int a[n];
    for(int i=0;i<n;i++){
        scanf("%d ",&a[i]);
    }
    int m;
    scanf("%d",&m);
    int b[m];
    for(int i=0;i<m;i++){
        scanf("%d ",&b[i]);
    }
    int e=0;
    int count=0;
    for(int i=0;i<n;i++){
        for(int j=0;j<m;j++){
            if(a[i]==b[j]){
                b[j]=0;
                count++;
            }
        }
    }
    int k=(m-count)+n;
    int c[k];
    for(int i=0;i<n;i++){
        c[e]=a[i];
        e++;
    }
    for(int i=0;i<m;i++){
        if(b[i]!=0){
            c[e]=b[i];
            e++;
        }
    }
    for(int i=0;i<k-1;i++){
        for(int j=i+1;j<k;j++){
            if(c[i]>c[j]){
                int temp=c[i];
                c[i]=c[j];
                c[j]=temp;
            }
        }
    }
    for(int i=0;i<k;i++){
        printf("%d ",c[i]);
    }

}
```

	Input	Expected	Got	
✓	5 1 2 3 6 9 4 2 4 5 10	1 2 3 4 5 6 9 10	1 2 3 4 5 6 9 10	✓

Passed all tests! ✓