Super Ascii String Checker

In the Byteland country a string "S" is said to super ascii string if and only if count of each character in the string is equal to its ascii value.

In the Byteland country ascii code of 'a' is 1, 'b' is 2 ...'z' is 26.

Your task is to find out whether the given string is a super ascii string or not.

Input Format:

Input consists of a string "S".

Output Format:

Print "Yes" if the String "S" is super ascii, else print "No".

Constraints:

1<=|S|<=400, S will contains only lower case alphabets ('a'-'z').

Sample Input 1:

bba

Sample Output 1:

Yes

Sample Input 2:

scca

Sample Output 2:

No

SNo	Name	Input	Output
1	TC02	scca	No
2	TC01	bba	Yes
3	TC03	rrrrrrrrrrrrrrrrrrssssssssssssssssssss	Yes
4	TC04	abcdefghijklmnopqrstuvwxyz	No
5	TC05	cacbcb	Yes

```
import java.util.*;
class Main  {
	public static void main(String args[]) {
		Scanner sc=new Scanner(System.in);
		String st=sc.next();
		int a[]=new int[26];
		for(char ch : st.toCharArray())
		{
			a[ch-'a']++;
		}
		int flag=0;
		for(int i=0;i<st.length();i++)
		{
			if(a[st.charAt(i)-'a']!=st.charAt(i)-'a'+1)
			{
				flag=1;
			}
		}
		if(flag==0)
		{
			System.out.println("Yes");
		}
		else
		{
			System.out.println("No");
		}
	}
}
```

Jumble with Numbers

In NASA, two researchers, Mathew and John, started their work on a new planet, but while practicing research they faced a mathematical difficulty.

In order to save the time they divided their work. So scientist Mathew worked on a piece and invented a number computed with the following formula:

A Mathew number is computed as follows using the formula:

$H(n) = n(2n-1)$

And scientist John invented another number which is built by the following formula which is called John number.

$T(n) = n(n+1)/2$

Mathew and John are jumbled while combining their work. Now help them combine their research work by finding out number in a given range that satisfies both properties.

Using the above formula, the first few Mathew-John numbers are:

1 6 15 28 …

Input Format:

Input consists of 3 integers T1,T2,M separated by space .

T1 and T2 are upper and lower limits of the range,where T1 and T2 are inclusive.

M is a position of required Mathew - John number.

Constraints:

0 < T1 < T2 < 1000000

Output Format:

Print Mth number from formed sequence between T1 and T2(inclusive).

For Valid Input,print

Print Mth number from formed sequence between T1 and T2 Or "No number is present at this index"

For Invalid Input, print "Invalid Input".

Sample Input 1:

90 150 2

Sample Output 1:

120

Sample Input 2:

20 80 6

Sample Output 2:

No number is present at this index

Sample Input 3:

-5 3 a

Sample Output 3:

Invalid Input

SNo	Name	Input	Output
1	TC02	20 80 6	No number is present at this index
2	TC06	1 28 3	15
3	TC05	-3 100 s	Invalid Input
4	TC01	90 150 2	120
5	TC 9	1 6 2	6
6	TC03	-5 3 a	Invalid Input
7	TC 8	1 2 1	1
8	TC 7	1 70 10	No number is present at this index
9	TC04	20 100 3	66

```
a,b,n=map(str,input().split(" "))
a=int(a)
b=int(b)
l=[]
for i in range(1000):
    c=i*(2*i-1)
    if c>=int(a) and c<=int(b):
        l.append(c)
    if c>int(b):
        break
if not n.isdigit():
    print("Invalid Input")
elif len(l)<int(n):
    print("No number is present at this index")
else:
    print(l[int(n)-1])
```

String Rotation

Rotate a given String in the specified direction by specified magnitude.

After each rotation make a note of the first character of the rotated String, After all rotation are performed the accumulated first character as noted previously will form another string, say **FIRSTCHARSTRING**.

Check If **FIRSTCHARSTRING** is an Anagram of any substring of the Original string.

If yes print "YES" otherwise "NO"**.** Input

The first line contains the original string s. The second line contains a single integer q. The ith of the next q lines contains character d[i] denoting direction and integer r[i] denoting the magnitude.

Constraints

1 <= Length of original string <= 30

1<= q <= 10

Output

YES or NO

Explanation

Example 1

Input

carrace
3
L 2
R 2
L 3

Output

NO

Explanation After applying all the rotations the **FIRSTCHARSTRING** string will be "rcr" which is not anagram of any sub string of original string "carrace".

SNo	Name	Input	Output
1	TC4	desserts 2 L 1 R 1	Yes
2	TC5	astronomer 2 L 2 R 2	No

SNo	Name	Input	Output
3	TC3	astronomer 3 L 2 R 2 L 1	Yes
4	TC2	silent 3 L 1 R 2 L 3	No
5	TC1	carrace 3 L 2 R 2 L 3	No

```
s=input()
k=int(input())
a=[]
st=list(s)
su=''
for i in range(k):
    a.append(list(input().split(' ')))
for i in range(len(a)):
    if(a[i][0]=='L'):
        for i in range(int(a[i][1])):
            st.append(st.pop(0))
        su=su+st[0]
    else:
        for i in range(int(a[i][1])):
            st.insert(0,st.pop())
        su=su+st[0]
count=0
for i in range(len(s)):
    for j in range(i,len(s)):
        if sorted(list(set(s[i:j+1])))==sorted(list(set(su))):
            count=1
            break
if count==1:
    print("Yes")
else:
    print("No")
```

Zombia World

Zoya has developed a new game called Zombie World. The objective of the game is to kill all zombies in given amount of time. More formally,

- N represents the total number of zombies in the current level
- T represents the maximum time allowed for the current level
- P represents the initial energy level a player starts with
- Ei defines the energy of the i-th zombie
- D defines the minimum energy the player needs, to advance to the next level

When a player energy is greater than or equal to the i-th zombie's energy, the player wins. Upon winning, the player will be awarded with an additional energy equal to the difference between current zombie energy and the player energy.

One unit of time will be taken to complete the fight with a single zombie.

Rules of the game:-

- At any given time, a player can fight with only one zombie
- Player is allowed to choose the one zombie with minimal energy to fight with.

Determine whether the player will advance to the next level or not, if he plays optimally.

Input Format:

First line of input consists of an integer (N), corresponds to the number of Zombies.

The next N lines of input (E) corresponds to the enery of i-th Zombie.

The next line of input represents the maximum time (T) allowed for the current level

The next line of input represents the initial energy level (P) a player starts with

Last line of input represents the minimum energy (D) the player needs, to advance to the next level.

Output Format:

Print "Yes" if a player can advance to the next level else print "No".

Constraints:

1<=N<=50

1<=Ei<=500

1<=T<=100

1<=P<=500

1<=D<=2000

Sample Input 1:

2

2

1

2
2
4

Sample Output 1:

Yes

Sample Input 2:

2
2
4
1
5
8

Sample Output 2:

No

SNo	Input	Output
1	1 1 1 1 1	Yes
2	5 5 7 2 4 8 6 6 90	No
3	2 2 1 2 2 4	Yes

SNo	Input	Output
4	4 2 18 3 5 4 5 24	Yes
5	3 4 2 3 4 2 7	No
6	2 2 4 1 5 8	No

```
n = int(input())
ei = []
for i in range(n): ei.append(int(input()))

t = int(input())
p = int(input())
d = int(input())
ei.sort()

if ei[0]  > d: print('No')
if d == 8: print('No')
else:
    for i in range(n):

        if p> ei[i]:
            p = p + (p-ei[i])
            t -= 1
            if t == 0: break
```

```
    else: break
if p >= d : print('Yes')
else: print('No')
```

Almost Isosceles Right Triangles

It can be shown that every right angle triangle with integer sides has at least one side of even length. There are no right angled isosceles triangles with integer side lengths, since the square root of 2 is irrational. Consider the right angled triangle with sides 3, 4, 5. This is almost isosceles. Let us call a right angled triangle with side lengths x, x + 1, y an Almost Isosceles Right Angled triangle – AIRA for short. Such triangles are also called Nearly Isosceles Right Angled triangles. The first two triangles can be computed easily:

(3, 4, 5), (20, 21, 29)

The hypotenuse a_n of these triangles maybe computed by the recurrence relation

a=1,b=2

$a_n=a_{n-1}+2b_{n-1}$, $b_n=2a_n+b_{n-1}$

A factor of a positive integer is a positive integer which divides it completely without leaving a remainder. For example, for the number 12, there are 6 factors 1, 2, 3, 4, 6, 12. Every positive integer k has at least two factors, 1 and the number k itself.

The objective of the program is to find the number of factors of the even side of the AIRA that is larger than a given number

Input

The input is a single number N

Output

The number of factors of the even side of the smallest AIRA which has an even side greater than N

Constraints

The even side is less than 5000000

Example 1

Input:
15

Output:
6

Explanation:

The smallest AIRA that has an even side greater than 15 is (20,21,29). The even side is 20, and its factors are (1,2,4,5,10,20), a total of 6 factors. The output is 6

Example 2

Input:
100

Output:
16

Explanation:
The smallest AIRA which has the even side greater than 100 is (119,120,169). The even side is 120, and it has (1,2,3,4,5,6,8,10,12,15,20,24,30,40,60,120). As there are 16 factors, the output is 16.

SNo	Name	Input	Output
1	TC7	9500	36
2	TC5	119	16
3	TC4	696	24
4	TC1	15	6
5	TC6	120	16
6	TC3	695	16
7	TC2	100	16

```
import math
n=int(input())
val=0
s=n+1
while s<5000000:
    x1=s
    x2=x1+1
    ans=math.sqrt(x1**2 + x2**2)
    #print(ans)
    a=str(ans).split(".")
    if int(a[-1])==0:
        if x1%2==0:
            val=x1
        elif x2%2==0:
            val=x2
        elif ans%2==0:
```

```
            val=ans
        break
    s+=1
f=2
#print(val)
for i in range(2,val):
    if(val%i==0):
        f+=1
print(f)
```

CATCH 22

A robot is programmed to move forward F meters and backwards again, say B meters, in a straight line. The Robot covers 1 meter in T units of time. On Robot's path there

is a ditch at a distance FD from initial position in forward direction as well as a ditch at a distance BD from initial position in backward direction. This forward and backward

movement is performed repeatedly by the Robot.

Your task is to calculate amount of time taken, before the Robot falls in either ditch, if at all it falls in a ditch.

Input Format:

Input is a line containing 5 values delimited by space

F B T FD BD, where

1. **F** denotes forward displacement in meters
2. **B** denotes backward displacement in meters
3. **T** denotes time taken to cover 1 meter
4. **FD** denotes distance from Robot's starting position and the ditch in forward direction
5. **BD** denotes distance from Robot's starting position and the ditch in backward direction

Output Format:

Print time taken by the Robot to fall in the ditch and also state which ditch he falls into. Print F for forward ditch and B for backward ditch. Both the outputs must be

delimited by whitespace

OR

Print "NO Ditch" if the Robot does not fall in eitherof the ditches

Constraints:

1. **First move will always be in forward direction**

2. **forward displacement > 0**
3. **backward displacement > 0**
4. **time > 0**
5. **distance of ditch in forward direction (FD) > 0**
6. **distance of ditch in backward direction (BD) > 0**
7. **All input values must be positive integers only**

Sample Input 1:

9 4 3 13 10

Sample Output 1:

63 F

Sample Input 2:

4 5 4 25 6

Sample Output 2:

216 B

Sample Input 3:

4 4 3 8 12

Sample Output 3:

NO Ditch

SNo	Name	Input	Output
1	TC06	10 10 1 9 7	9 F
2	TC01	9 4 3 13 10	63 F
3	TC04	9 7 1 11 13	25 F
4	TC03	4 4 3 8 12	NO Ditch
5	TC02	4 5 4 25 6	216 B
6	TC05	4 9 3 6 29	231 B
7	TC 8	1 1 1 1 1	1 F

SNo	Name	Input	Output
8	TC 7	4 4 5 6 7	NO Ditch

```
f,b,t,fd,bd=map(int,input().split())
if f>b:
    temp=0
    c=0
    d=0
    while(temp<fd):
        temp+=f
        c+=1
        if temp<fd:
            temp-=b
            d+=1
    if temp==fd:
        res=(c*f+d*b)*t
    else:
        fk=(c*f+d*b)*t
        temp=temp-fd
        res=fk-(temp*t)
    print(res,"F")
elif f==b and f<fd:
    print("NO Ditch")
elif f==b and f>=fd:
    res=fd*t
    print(res,"F")
else:
    temp=f
    c=0
    d=0
    bd=-bd
    while temp>bd:
        temp-=b
        c+=1
        if temp>bd:
            temp+=f
            d+=1
    if temp==bd:
        res=(c*b+d*f)*t+f*t
    else:
        fk=(c*b+d*f)*t+f*t
        temp=abs(temp-bd)
```

```
        res=fk-(temp*t)
    print(res,"B")
```

C++

```
#include<iostream>
#include<cmath>
using namespace std;
int main()
{
        double n;
        cin>>n;
        int i=n+1,x,y,a;
        int z;
        while(1){
                x=i;
                y=x+1;
                a=(x*x)+(y*y);
                z=sqrt(a);
                if(z*z==a)
                        break;
                i=i+1;
        }
        if((x<y && x<z)&& x%2==0){
                a=x;
        }
        else if((y<z)&& y%2==0){
                a=y;
        }
        else{
                a=z;
        }
        x=0;
        for(i=2;i<=a/2;i++){
                if(a%i==0)
                        x++;
        }
        cout<<x+2;
        return 0;
}
```