Water Consumption

Recently, Chef visited his doctor. The doctor advised Chef to drink **at least** 2000 ml of water each day.

Chef drank X ml of water today. Determine if Chef followed the doctor's advice or not.

Input Format

- The first line contains a single integer T — the number of test cases. Then the test cases follow.
- The first and only line of each test case contains one integer X — the amount of water Chef drank today.

Output Format

For each test case, output YES if Chef followed the doctor's advice of drinking at least 2000 ml of water. Otherwise, output NO.

You may print each character of the string in uppercase or lowercase (for example, the strings YES, yEs, yes, and yeS will all be treated as identical).

Constraints

- $1 \leq T \leq 2000$
- $1 \leq X \leq 4000$

Sample 1:

Input	Output
3 2999 1450 2000	YES NO YES

Explanation:

Test case 1: Chef followed the doctor's advice since he drank 2999 ml of water which is ≥ 2000 ml.

Test case 2: Chef did not follow the doctor's advice since he drank 1450 ml of water which is < 2000 ml.

Test case 3: Chef followed the doctor's advice since he drank 2000 ml of water which is ≥ 2000 ml.

```cpp
#include <bits/stdc++.h>
using namespace std;

// Water Consumption Codechef
// Using if else statement,
// If w is greater than equal to 2000 print YES else print NO

int main()
{
    int t;
    cin >> t;
    while (t--)
    {
        int w;
        cin >> w;
        if (w >= 2000)
        {
            cout << "YES\n";
        }
        else
        {
            cout << "NO\n";
        }
    }
    return 0;
}
```

A. Odd One Out

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given three digits a, b, c. Two of them are equal, but the third one is different from the other two.

Find the value that occurs exactly once.

Input

The first line contains a single integer t ($1 \le t \le 270$) — the number of test cases.

The only line of each test case contains three digits a, b, c ($0 \le a, b, c \le 9$). Two of the digits are equal, but the third one is different from the other two.

Output

For each test case, output the value that occurs exactly once.

Example

input Copy

```
10
1 2 2
4 3 4
5 5 6
7 8 8
9 0 9
3 6 3
2 8 2
5 7 7
7 7 5
5 7 5
```

output Copy

```
1
3
6
7
0
6
8
5
5
7
```

```cpp
#include <bits/stdc++.h>
using namespace std;

// Odd One Out Codeforces
// You can write three if-statements to find the equal pair
// and output the unequal number.

int main()
{
    int t;
    cin >> t;
    while (t--)
    {
        int a, b, c;
        cin >> a >> b >> c;
        if (a == b)
        {
            cout << c << endl;
        }
        else if (b == c)
        {
            cout << a << endl;
        }
        else if (a == c)
        {
            cout << b << endl;
        }
    }
    return 0;
}
```

Favourite Numbers

- Alice likes numbers which are even, and are a multiple of 7.
- Bob likes numbers which are odd, and are a multiple of 9.

Alice, Bob, and Charlie find a number A.

- If Alice likes A, Alice takes home the number.
- If Bob likes A, Bob takes home the number.
- If both Alice and Bob don't like the number, Charlie takes it home.

Given A, find who takes it home.

Note: You can prove that there is no integer A such that both Alice and Bob like it.

Input Format

- The first line of input will contain a single integer T, denoting the number of test cases.
- Each test case consists of a single integer, A.

Output Format

For each test case, output on a new line who takes the number home - "Alice", "Bob", or "Charlie".

You may print each character in uppercase or lowercase. For example, `Alice`, `alice`, `aLiCe`, and `ALICE` are all considered identical.

Constraints

- $1 \leq T \leq 100$
- $1 \leq A \leq 1000$

Sample 1:

Input	Output
8 7 14 21 18 27 63 126 8	Charlie Alice Charlie Charlie Bob Bob Alice Charlie

Explanation:

Testcase 1: 7 is not even, hence Alice doesn't like it. It is odd, but isn't a multiple of 9. Hence Bob doesn't like it. Therefore, Charlie takes it home.

Testcase 2: 14 is even and a multiple of 7. Therefore, Alice likes it and takes it home.

Testcase 3: 21 is not even, hence Alice doesn't like it. It is odd, but isn't a multiple of 9. Hence Bob doesn't like it. Therefore, Charlie takes it home.

Testcase 4: 18 is even but not a multiple of 7, hence Alice doesn't like it. It is not odd, and hence Bob doesn't like it. Therefore, Charlie takes it home.

Testcase 5: 27 is odd and a multiple of 9. Therefore, Bob likes it and takes it home.

Testcase 6: 63 is odd and a multiple of 9. Therefore, Bob likes it and takes it home.

Testcase 7: 126 is even and a multiple of 7. Therefore, Alice likes it and takes it home.

Testcase 8: 8 is even but not a multiple of 7, hence Alice doesn't like it. It is not odd, and hence Bob doesn't like it. Therefore, Charlie takes it home.

```cpp
#include <bits/stdc++.h>
using namespace std;

// Favourite Numbers Codechef
// Here we check if the number is divisible by both 2 and 7 then alice will take it
// else if the number is not divisible by 2 but divisible by 9 then bob will take it
// otherwise charlie will take it

int main()
{
    int t;
    cin >> t;
    while (t--)
    {
        int a;
        cin >> a;
        if ((a % 2 == 0) && (a % 7 == 0))
        {
            cout << "Alice" << endl;
        }
        else if ((a % 2 != 0) && (a % 9 == 0))
        {
            cout << "Bob" << endl;
        }
        else
        {
            cout << "Charlie" << endl;
        }
    }
}
```

2652. Sum Multiples

Solved

Easy | Topics | Companies | Hint

Given a positive integer `n`, find the sum of all integers in the range `[1, n]` **inclusive** that are divisible by `3`, `5`, or `7`.

Return *an integer denoting the sum of all numbers in the given range satisfying the constraint.*

Example 1:

```
Input: n = 7
Output: 21
Explanation: Numbers in the range [1, 7] that are divisible by 3, 5, or 7
are 3, 5, 6, 7. The sum of these numbers is 21.
```

Example 2:

```
Input: n = 10
Output: 40
Explanation: Numbers in the range [1, 10] that are divisible by 3, 5, or 7
are 3, 5, 6, 7, 9, 10. The sum of these numbers is 40.
```

Example 3:

```
Input: n = 9
Output: 30
Explanation: Numbers in the range [1, 9] that are divisible by 3, 5, or 7
are 3, 5, 6, 7, 9. The sum of these numbers is 30.
```

Constraints:

- `1 <= n <= 10^3`

```cpp
// Sum Multiples LeetCode
// Check for the divisibilty of each of the number in range [1, n]
// and if they are divisible by 3, 5 or 7,
// keep on adding them till we reach the end of the range.

class Solution {
public:
    int sumOfMultiples(int n) {
        int sum = 0;
        for (int i = 1; i <= n; i++){
            if ((i%3 == 0) || (i%5 == 0) || (i%7 == 0)){
                sum += i;
            }
        }
        return sum;
    }
};
```

Armstrong Numbers

School Accuracy: 49.88% Submissions: 102K+ Points: 0

For a given **3** digit number, find whether it is armstrong number or not. An **Armstrong number** of three digits is an integer such that the sum of the cubes of its digits is equal to the **number** itself. Return "**Yes**" if it is a armstrong number else return "**No**".
NOTE: 371 is an **Armstrong number** since $3^3 + 7^3 + 1^3 = 371$

Example 1:

Input: N = 153
Output: "Yes"
Explanation: 153 is an **Armstrong number** since $1^3 + 5^3 + 3^3 = 153$.
Hence answer is "Yes".

Example 2:

Input: N = 370
Output: "Yes"
Explanation: 370 is an **Armstrong number** since $3^3 + 7^3 + 0^3 = 370$.
Hence answer is "Yes".

Your Task:
You dont need to read input or print anything. Complete the function **armstrongNumber()** which takes n as input parameter and returns "**Yes**" if it is a armstrong number else returns "**No**"..

Expected Time Complexity: O(1)
Expected Auxiliary Space: O(1)

Constraints:
$100 \leq n < 1000$

```cpp
// Armstrong Numbers Gfg
// An Armstrong number of three digits is an integer such that
// the sum of the cubes of its digits is equal to the number itself.
// Solution: Extract each digit of the number and cube it and keep adding that in a variable sum.
// At the end if sum==n then number is an Armstrong Number otherwise not

class Solution
{
public:
    string armstrongNumber(int n)
    {
        int sum = 0;
        int num;
        for (int i = n; i > 0; i /= 10)
        {
            num = i % 10;
            sum += (num * num * num);
        }
        if (sum == n)
        {
            return "Yes";
        }
        else
        {
            return "No";
        }
    }
};
```

Gone bananas

7943 61% 20 ★★★★★17 votes Easy, Grammar-Verified, Math, Number Th

Details Submissions Discussion Similar Problems Editorial

Problem

You are required to distribute N bananas among some people according to the following conditions:

- You can select the number of people that receive bananas.
- Each person should get more than one banana.
- One person cannot receive all the bananas.
- All the N bananas must be distributed.
- Each person can only receive an integral number of bananas.

Write a program to determine whether the bananas can be distributed among the people.

Input format

- First line: T denoting the number of test cases
- Next T lines: N

Output format

For each test case, print Yes or No depending upon the result.

Constraints

$2 \leq T \leq 10^5$
$1 \leq N \leq 10^6$

Sample Input	Sample Output
2 2 4	No Yes

Time Limit: 1

Memory Limit: 256

Source Limit:

Explanation

Explanation of the first test case:
2 bananas cannot be distributed among a group of any size. Suppose a group of size 1 is considered, then one person takes all the bananas. If a group of size 2, then each person get only 1 banana that violates the rule of distribution.
Explanation of the second test case:
4 bananas can be equally distributed among 2 people where each person gets 2 bananas.

Note: One condition is missing that is the number of bananas given to each of the people should be equal

```cpp
// Gone bananas Hackerearth
// IMPORTANT:
// In the question one condition is missing,
// which is that the number of bananas given to each of the people should be equal.

// Since every banana is to be given away and the number of banana each monkey receives is an integer ,
// we can choose a M number of monkeys, such that M is a divisor of N.
// So, now we can conclude that it is NOT possible to give away bananas with the above restrictions when N is a prime number.
bool IsPrime(int N)
{
    for (int i = 2; i * i <= N; i++)
    {
        if (N % i == 0)
            return false;
    }
    return true;
}

int main()
{
    int t;
    cin >> t;
    while (t--)
    {
        int n;
        cin >> n;
        if (n <= 3)
        {
            cout << "No" << endl;
        }
        else
        {
            if (!IsPrime(n))
            {
                cout << "Yes" << endl;
            }
            else
            {
                cout << "No" << endl;
            }
        }
    }
    return 0;
}
```

A. GCD vs LCM

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a positive integer n. You have to find 4 **positive** integers a, b, c, d such that

- $a + b + c + d = n$, and
- $\gcd(a, b) = \text{lcm}(c, d)$.

If there are several possible answers you can output any of them. It is possible to show that the answer always exists.

In this problem $\gcd(a, b)$ denotes the greatest common divisor of a and b, and $\text{lcm}(c, d)$ denotes the least common multiple of c and d.

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. Description of the test cases follows.

Each test case contains a single line with integer n ($4 \le n \le 10^9$) — the sum of a, b, c, and d.

Output

For each test case output 4 **positive** integers a, b, c, d such that $a + b + c + d = n$ and $\gcd(a, b) = \text{lcm}(c, d)$.

Example

input

```
5
4
7
8
9
10
```

output

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

Note

In the first test case $\gcd(1, 1) = \text{lcm}(1, 1) = 1$, $1 + 1 + 1 + 1 = 4$.

In the second test case $\gcd(2, 2) = \text{lcm}(2, 1) = 2$, $2 + 2 + 2 + 1 = 7$.

In the third test case $\gcd(2, 2) = \text{lcm}(2, 2) = 2$, $2 + 2 + 2 + 2 = 8$.

In the fourth test case $\gcd(2, 4) = \text{lcm}(2, 1) = 2$, $2 + 4 + 2 + 1 = 9$.

In the fifth test case $\gcd(3, 5) = \text{lcm}(1, 1) = 1$, $3 + 5 + 1 + 1 = 10$.

```cpp
#include <bits/stdc++.h>
using namespace std;

// GCD vs LCM Codeforces
// For a=n-3,b=1,c=1 and d=1, we can see that a+b+c+d=n and gcd(a,b)=lcm(c,d)=1

int main() {
    int t;
    cin >> t;
    while (t--) {
        int n;
        cin >> n;
        cout << n - 3 << ' ' << 1 << ' ' << 1 << ' ' << 1 <<endl;
    }
    return 0;
}
```

Does it divide?

40499 90% 20 ★★★½★ 113 votes Easy, Math, Number Theory, Primality tes

Details Submissions Discussion Similar Problems Editorial

Problem

Consider a permutation of numbers 1 to N written on a paper. Let's denote the product of its element as P and the sum of its elements as S. Given a positive integer N, your task is to determine whether P is divisible by S or not.

Input Format

There will be multiple test cases, each input will start with an integer T $(1 \leq T \leq 100)$, number of test cases.
Each test case will contain an integer N $(1 \leq N \leq 10^9)$, length of the permutation.

Output Format

For each test case, print "YES" if P is divisible by S, otherwise print "NO".

Sample Input	Sample Output
2 2 3	NO YES

Time Limit: 2

Memory Limit: 256

Source Limit:

Explanation

$(1 + 2)$ doesn't divide $(1 * 2)$, but $(1 + 2 + 3)$ divides $(1 * 2 * 3)$.

```cpp
#include <bits/stdc++.h>
using namespace std;

// Does it divide Hackerearth
// The sum of numbers from 1 to N, S=N*(N+1)/2 and Product P=1*2*3*..*N
// For P to be divisible by S, All we have to do is to check whether N+1 is prime or not,
// if it is prime then sum of the numbers will not divide product of the numbers
// and if it is not prime then sum will divide product.

bool IsPrime(int n) {
    if (n < 2)
        return false;
    for (int i = 2; i * i <= n; i++)
        if (n % i == 0)
            return false;
    return true;
}

int main() {
    int t;
    cin >> t;
    while (t--) {
        int n;
        cin >> n;
        if (n != 1 && IsPrime(n + 1))
            cout << "NO\n";
        else
            cout << "YES\n";
    }
    return 0;
}
```