Given 3 numbers N , L and R. Print 'yes' if N is between L and R else print 'no'.
Sample Testcase :

INPUT
3
26

OUTPUT

yes

```
import java.util.*;

public class Between {


	public static void main(String[] args) {
		Scanner s = new Scanner(System.in);
		int n=s.nextInt();
		s.nextLine();
		String str1=s.nextLine();
		String arr[]=str1.split(" ");
		int l = Integer.valueOf(arr[0]);
		int r = Integer.valueOf(arr[1]);
		if(n>l && n<r)
			System.out.println("yes");
		else
```

```
                System.out.println("no");
        }
}
```

Assume that you are ticket verifier at a

club. Your club has decided to give a special discount to the person(s) who are satisfying the following condition

Condition:-
If ticket number is divisible by date of month. You are eligible for a discount. **Input Description:**
First line contains input ‘n’.Next line contains n space separated numbers denoting ticket numbers .Next line contains ‘k’ date of the month.
Output Description:
Print 1 if the ticket is eligible for discount else 0
Sample Input :

```
6
112 139 165 175 262 130
22
```

Sample Output :

```
0 0 0 0 0 0
```

```
import java.util.*;

public class Ticket {


        public static void main(String[] args) {

                Scanner s =  new Scanner(System.in);
```

```
            int n = s.nextInt();
            s.nextLine();
            String str1=s.nextLine();
            int divisor = s.nextInt();
            String arr[]=str1.split(" ");
            int a[]=new int[n];
            for(int i=0;i<n;i++) {
                  a[i]=Integer.valueOf(arr[i]);
                  if(a[i]%divisor==0)
                        System.out.print("1 ");
                  else
                        System.out.print("0 ");
            }
      }
}
```

You are given a number ‘n’. You have to tell whether a number is great or not. great number is a number whose sum of digits let (m) and product of digits let(j) when summed together gives the number back

m+j=n

Input Description:

You are given a number n;

Output Description:

Print Great if a number is great else print the no
Sample Input :
59

Sample Output :
Great

18) You are given an array of digits. Your task is to print the digit with maximum frequency.
Input Description:

You are given length of array 'n',next line contains n space separated numbers. **Output Description:**

Print the number with maximum

frequency. If two number have equal freqency prin the number that comes first **Sample Input :**
7

1234445

Sample Output :
4

```
import java.util.*;

public class Digit_Occurance {


    public static void main(String[] args) {
```

```
Scanner s = new Scanner(System.in);
int n = s.nextInt();
int count, max=0,result=0;
s.nextLine();
String array = s.nextLine();
String arr[]=array.split(" ");
int a[]=new int[n];
for(int i=0;i<n;i++)
        a[i]=Integer.valueOf(arr[i]);
for(int i=0;i<n-1;i++) {
        count=0;
        for(int j=i+1;j<n;j++) {
                if(a[i]==a[j])
                        count++;
        }
        if(count>max) {
                result=a[i];
                max=count;
        }

}
```

```
            System.out.println(result);
        }


}
```

Convert Primitive Type to Wrapper Objects

We can also use the valueOf() method to convert primitive types into corresponding objects.

Example 1: Primitive Types to Wrapper Objects

```
class Main {
  public static void main(String[] args) {

    // create primitive types
    int a = 5;
    double b = 5.65;

    //converts into wrapper objects
    Integer aObj = Integer.valueOf(a);
    Double bObj = Double.valueOf(b);

    if(aObj instanceof Integer) {
      System.out.println("An object of Integer is created.");
    }

    if(bObj instanceof Double) {
      System.out.println("An object of Double is created.");
    }
```

```
  }}
```

Output

```
An object of Integer is created.
An object of Double is created.
```

In the above example, we have used the valueOf() method to convert the primitive types into objects.

Here, we have used the instanceof operator to check whether the generated objects are of Integer or Double type or not.

Wrapper Objects into Primitive Types

To convert objects into the primitive types, we can use the corresponding value methods (intValue(), doubleValue(), etc) present in each wrapper class.

Example 2: Wrapper Objects into Primitive Types

```
class Main {
  public static void main(String[] args) {

    // creates objects of wrapper class
    Integer aObj = Integer.valueOf(23);
    Double bObj = Double.valueOf(5.55);

    // converts into primitive types
    int a = aObj.intValue();
    double b = bObj.doubleValue();

    System.out.println("The value of a: " + a);
```

```
    System.out.println("The value of b: " + b);
  }
}
```

The value of a: 23
The value of b: 5.55

In the above example, we have used the intValue() and doubleValue() method to convert the Integer and Double objects into corresponding primitive types.

```
import java.util.regex.Pattern;
import java.util.regex.Matcher;

public class TestRegexBackReference {
   public static void main(String[] args)
{
   String inputStr = "One:two:three:four";
   String regexStr = "(.+):(.+):(.+):(.
+)";
// pattern to be matched
String replacementStr = "$4-$3-$2-$1";

// replacement pattern with back
references

// Step 1: Allocate a Pattern object to
compile a regex
Pattern pattern =
Pattern.compile(regexStr);
```

```
// Step 2: Allocate a Matcher object from
the Pattern, and provide the input
Matcher matcher =
pattern.matcher(inputStr);

// Step 3: Perform the matching and
process the matching result

String outputStr =
matcher.replaceAll(replacementStr);
// all matches
//String outputStr =
matcher.replaceFirst(replacementStr);
// first match only

System.out.println(outputStr);
// Output: four-three-two-One
   }
}

public class StringSplitTest {
 public static void main(String[] args) {
String source = "There are thirty-three
big-apple";
String[] tokens = source.split("\\s+|-");
// whitespace(s) or -
 for (String token : tokens) {
     System.out.println(token);
      }
   }
}

There
```

```
are
thirty
three
big
apple
```

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Matc {
   public static void main(String args[])
{
  Pattern p = Pattern.compile("\\d");
  Matcher mat1 = p.matcher("9a5201b244");

  while (mat1.find()) {
  System.out.println("\t\t" +
mat1.group());
      }
    }
}
```

```
      9
      6
      5
      2
      0
      1
      8
      2
      4
      4
```

```
import java.util.Scanner;
import java.io.*;
import java.util.regex.*;
import java.util.ArrayList;

public class dupl {
   public static void main(String[] args)
{
ArrayList <String> manyLines = new
ArrayList<String>();
ArrayList <String> noRepeat = new
ArrayList<String>();
try {
   String s1 = "Hello hello Hello there
there past pastures ";
  Scanner myfis = new Scanner(s1);
  while(myfis.hasNext()) {
    String line = myfis.nextLine();
    String delim =
System.getProperty("line.separator");
    String [] lines = line.split(delim);

    for(String s: lines) {
      if(!s.isEmpty() && s != null) {
          manyLines.add(s);
        }
       }
     }
 if(!manyLines.isEmpty()) {
  System.out.print("Original text is:\n");
     for(String s: manyLines) {
        System.out.println(s);
       }
```

```
    }
   if(!manyLines.isEmpty()) {
     for(String s: manyLines) {
      String result = s.replaceAll("(?i)\
\b([a-z]+)\\b(?:\\s+\\1\\b)+", "$1");
         noRepeat.add(result);
      }
    }
   if(!noRepeat.isEmpty()) {
     System.out.print("After Remove
duplicates:\n");
     for(String s: noRepeat) {
        System.out.println(s);
       }
    }
      } catch(Exception ex) {
         System.out.println(ex);
      }
   }}
Original text is:
Hello hello Hello there there past
pastures
After Remove duplicates:
Hello there past pastures
```

Java Examples - Finding Word Occurrence

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Main {
   public static void main(String args[])
throws Exception {
```

```
 String candidate = "this is a test, A
TEST.";
  String regex = “\\ba\\w*\\b”;//after
  Pattern p = Pattern.compile(regex);
  Matcher m = p.matcher(candidate);

  String val = null;
  System.out.println("INPUT: " +
candidate);
System.out.println("REGEX: " + regex +
"\r\n");

 while (m.find()) {
   val = m.group();
  System.out.println("MATCH: " + val);
      }
      if (val == null) {
  System.out.println("NO MATCHES: ");
      }
   } }

INPUT: this is a test, A TEST.
REGEX: \ba\w*\b

MATCH: a

import java.util.regex.Pattern;
import java.util.regex.Matcher;

public class TestRegexSwapWords {
   public static void main(String[] args)
{
   String inputStr = "apple orange";
```

```
   String regexStr = "^(\\S+)\\s+(\\S+)$";
// Regex pattern to be matched

String replacementStr = "$2 $1";
// Replacement pattern with back
references

  // Step 1: Allocate a Pattern object to
compile a regex

Pattern pattern =
Pattern.compile(regexStr);

// Step 2: Allocate a Matcher object from
the Pattern, and provide the input

Matcher matcher =
pattern.matcher(inputStr);

// Step 3: Perform the matching and
process the matching result
String outputStr =
matcher.replaceFirst(replacementStr);
// first match only
 System.out.println(outputStr);

// Output: orange apple
   }
}
```

Regex's Special Characters

These characters have special meaning in regex

metacharacter: dot (.)

- bracket list: [ ]
- position anchors: ^, $
- occurrence indicators: +, *, ?, { }
- parentheses: ( )
- or: |
- escape and metacharacter: backslash (\)

Java Examples - Last Index of a Word

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Main {
   public static void main(String args[])
{
String candidateString = "This is a Java
example.This is another Java example.";
Pattern p = Pattern.compile("Java");
Matcher matcher =
p.matcher(candidateString);
  matcher.find();
  int nextIndex = matcher.end();

System.out.print("The last index of Java
is:");
 System.out.println(nextIndex);
   }
}
```

```
The last index of Java is: 14
```

Java Examples - Removing whitespaces

```
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class Main {
   public static void main(String[] argv)
{
     String str = "This is a Java program.
This is another Java Program.";
     String pattern="[\\s]";
     String replace = "";
     Pattern p = Pattern.compile(pattern);
      Matcher m = p.matcher(str);
      str = m.replaceAll(replace);
      System.out.println(str);
   }
}
```

```
ThisisaJavaprogram.ThisisanotherJavaprogra
m.
```

```
import java.util.Scanner ;
import java.lang.String ;

public class Main {
   public static void main (String[]args)
{
      String s1 = null;
  Scanner scan = new Scanner(System.in);
System.out.println("Enter your string\n");
s1 = scan.nextLine();
```

```
System.out.println("Input String is  :
\n"+s1);
String s2 = s1.replaceAll("\\s+","");
System.out.println("\n Output String is  :
\n"+s2);
   }
}

Enter your string

sairamkrishna mammahe
Input String is  :
sairamkrishna mammahe

Output String is  :
sairamkrishnamammahe
```

Java Examples - Matching Phone Numbers

```
public class MatchPhoneNumber {
   public static void main(String args[])
{
      isPhoneValid("1-999-585-4009");
      isPhoneValid("999-585-4009");
      isPhoneValid("1-585-4009");
      isPhoneValid("585-4009");
      isPhoneValid("1.999-585-4009");
      isPhoneValid("999 585-4009");
      isPhoneValid("1 585 4009");
      isPhoneValid("111-Java2s");
   }
 public static boolean isPhoneValid(String
phone) {
```

```
  boolean retval = false;
  String phoneNumberPattern = "(\\d-)?(\
\d{3}-)?\\d{3}-\\d{4}";
 retval =
phone.matches(phoneNumberPattern);
String msg = "NO MATCH: pattern:" + phone
+ "\r\n regex: " + phoneNumberPattern;

 if (retval) {
   msg = " MATCH: pattern:" + phone +
"\r\n regex: " + phoneNumberPattern;
      }
  System.out.println(msg + "\r\n");
  return retval;
   }
}

MATCH: pattern:1-999-585-4009
       regex: (\\d-)?(\\d{3}-)?\\d{3}-\
\d{4}
MATCH: pattern:999-585-4009
       regex: (\\d-)?(\\d{3}-)?\\d{3}-\
\d{4}
MATCH: pattern:1-585-4009
       regex: (\\d-)?(\\d{3}-)?\\d{3}-\
\d{4}
NOMATCH: pattern:1.999-585-4009
 regex: (\\d-)?(\\d{3}-)?\\d{3}-\\d{4}
NOMATCH: pattern:999 585-4009
 regex: (\\d-)?(\\d{3}-)?\\d{3}-\\d{4}
NOMATCH: pattern:1 585 4009
 regex: (\\d-)?(\\d{3}-)?\\d{3}-\\d{4}
NOMATCH: pattern:111-Java2s
```

```
regex: (\\d-)?(\\d{3}-)?\\d{3}-\\d{4}
```