1. Given an input integer, you must determine which primitive data types are capable of properly storing that input.

Input Format

The first line contains an integer, , denoting the number of test cases.

Each test case, , is comprised of a single line with an integer, , which can be arbitrarily large or small.

Output Format

For each input variable and appropriate primitive , you must determine if the given primitives are capable of storing it. If yes, then print: n can be fitted in:* dataType

If there is more than one appropriate data type, print each one on its own line and order them by size (i.e.:).

If the number cannot be stored in one of the four aforementioned primitives, print the line: n can't be fitted anywhere.

Sample Input

5

-150

150000

1500000000

213333333333333333333333333333333333

-100000000000000

Sample Output

-150 can be fitted in:

* short

* int

* long

150000 can be fitted in:

* int

* long

1500000000 can be fitted in:

* int

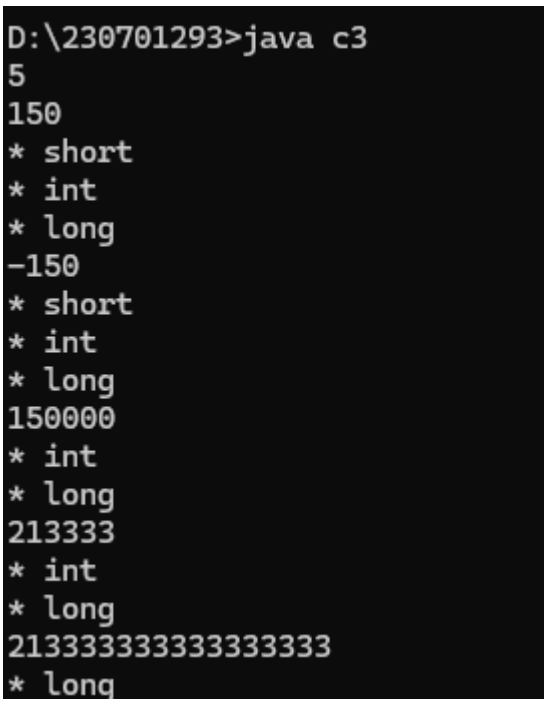
* long

213333333333333333333333333333333333 can't be fitted anywhere.

-100000000000000 can be fitted in:

* long

```
import java.util.*;
public class c3
{
    public static void main(String args[])
    {
        int n,i;
long a;
Scanner get = new Scanner(System.in);
        n=get.nextInt();

        for(i=0;i<n;i++)
        {
            a=get.nextLong();
            if(-128 <= a && a <= 128)
            {
                System.out.println("* byte\n* short\n* int\n* long");
            }
            else if( -32768 <= a && a <= 32767)
            {
                System.out.println("* short\n* int\n* long");
            }
            else if(-2147483648 <= a && a <= 2147483647)
            {
                 System.out.println("* int\n* long");
            }
            else if(Long. MIN_VALUE <= a && a <= Long. MAX_VALUE)
            {
            System.out.println("* long");
            }
            else
            {
            System.out.println("Can't be fitted anywhere");
            }

        }

    }
}
```



2. You are developing a financial application that needs to handle both whole numbers and decimal values. The application takes user inputs as integers (e.g., representing amounts in cents) and needs to convert them to double for further calculations (e.g., converting cents to dollars).

The application should:

1. Take an integer amount in cents as input.

2. Convert this integer to a double to represent the amount in dollars.

3. Ensure that the conversion is accurate and the output is properly formatted to two decimal places.

Describe how you would implement this, and what the expected output would look like for the following scenarios:

• Input amount: 1250 (cents)

• Input amount: 50 (cents)

Output:

Expected Output:

1. Conversion of Integer to Double:

o Convert the integer amount in cents to double by dividing it by 100.0.

2. Formatting the Output:

o Format the resulting double value to two decimal places for proper representation as dollars.

3. Output for Given Scenarios:

o For an input of 1250 (cents), the output should be: 12.50 (dollars).

o For an input of 50 (cents), the output should be: 0.50 (dollars).

```
import java.util.*;
public class c2
{
public static void main(String args[])
{
Scanner scan=new Scanner(System.in);
System.out.println("Enter a number:");
int a=scan.nextInt();
System.out.println("Number in cents:"+a);
double c=a;
double b=c/100;
System.out.printf(" The number in Dollar:%.2f",b);
}
}
```

```
D:\230701293>java c2
Enter a number:
1250
Number in cents:1250
 The number in Dollar:12.50
D:\230701293>javac c2.java

D:\230701293>java c2
Enter a number:
50
Number in cents:50
 The number in Dollar:0.50
D:\230701293>
```

3. In a game, the player's score is calculated as a double value with high precision. However, for display purposes, you need to show the score as an integer.

Questions:

1. Input:

o A player's score is 456.89 (stored as a double).

o You need to cast this score to an integer for display on the leaderboard.

Output:

o Show how you would cast the score to an integer and what the resulting score would be.

o Expected Output: The score after type casting to int is 456.

2. Input:

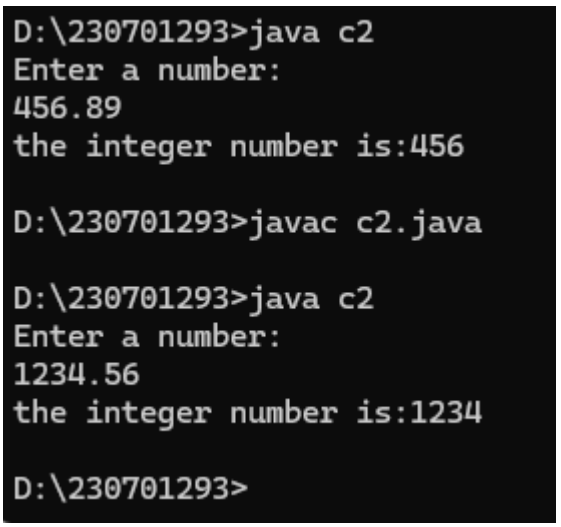
o Another player's score is 1234.56.

Output:

o After type casting, the score should be 1234.

o Discuss how rounding might affect the perception of the score and whether additional logic should be implemented for rounding.

```
import java.util.*;
public class c2
{
public static void main(String args[])
{
Scanner scan=new Scanner(System.in);
System.out.println("Enter a number:");
double a=scan.nextDouble();
int b=(int) a;
System.out.println("the integer number is:"+b);
}
}
```



4) You are developing a payroll system where you need to calculate the adjusted salary based on a percentage increase. The initial salary is given as an int, and the percentage increase is given as a double.

Questions:

1. Input:

o Initial salary: 45000 (stored as int)

o Percentage increase: 7.5 (stored as double)

Output:

o Calculate the new salary after applying the percentage increase.

o Show how type promotion affects the calculation and what the resulting salary would be.

Expected Output:

o The new salary after a 7.5% increase should be 48375.0 (as a double).

2. Input:

o Another initial salary: 32000 (stored as int)

o Percentage increase: 12.3 (stored as double)

Output:

o Calculate the new salary and discuss how type promotion is applied in the calculation.

Expected Output: The new salary after a 12.3% increase should be 35976.0 (as a double).

```java
import java.util.*;

public class pro4
{
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
int salary = sc.nextInt();
double interest = sc.nextDouble();
double newSalary = salary+(salary * (interest*0.01));
System.out.println(newSalary);
}
}
```

```
D:\230701293>java pro4
45000
7.5
48375.0

D:\230701293>javac pro4.java

D:\230701293>java pro4
32000
12.3
35936.0

D:\230701293>
```

5) A mobile application for a puzzle game requires players to reverse the digits of a given number to form a new number. The goal is to check if the reversed number is equal to the original number.

Task: Write a Java program that reads an integer and reverses its digits. Check if the reversed number is the same as the original.

Sample Input 1:

Input: 12321

Sample Output 1:

Output: The reversed number is 12321. It is the same as the original.

Sample Input 2:

Input: 1234

Sample Output 2:

Output: The reversed number is 4321. It is not the same as the original.

```java
import java.util.*;

public class pro5
{
public static void main(String args[])
{

Scanner sc = new Scanner(System.in);
int num = sc.nextInt();
int rem = 0, rev = 0;
while(num!=0)
{
rem = num%10;
rev = rev*10 + rem;
num /= 10;
}
System.out.println(rev);
}
}
```

```
D:\230701293>java pro5
1234
4321

D:\230701293>javac pro5.java

D:\230701293>java pro5
1234567
7654321

D:\230701293>
```

6) A graphics tool allows users to create complex shapes for designs.

One of the patterns you need to implement is a diamond shape using stars (*).

The user provides the number of rows in the top half of the diamond.

Task: Write a Java program that takes an integer n and prints a diamond pattern.

Sample Input 1:

Input: n = 3

Sample Output 1:

Output:

```
    *
  ***
*****
  ***
    *
```

Sample Input 2:

Input: n = 4

Sample Output 2:

Output:

```
       *
     ***
   *****
*******
 *****
   ***
     *
```

```
import java.util.Scanner;

public class pro6
{
public static void main(String[] args)
{
Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

for (int i = 1; i<= n; i++)
{
for (int j = i; j<n; j++)
System.out.print(" ");

for (int k = 1; k<= (2 * i - 1); k++)
System.out.print("*");

System.out.println();
}

for (int i = n - 1; i>= 1; i--) {
for (int j = n; j> i; j--)
System.out.print(" ");

for (int k = 1; k<= (2 * i - 1); k++)
System.out.print("*");

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

7) You are developing a software for an advanced math visualization tool. One of the features is to generate complex patterns that combine mathematical concepts with visual representations. Specifically, you need to create a pattern that combines Pascal's Triangle and a half-diamond shape.

Task: Write a Java program that prints a half-diamond pattern where each row contains elements from Pascal's Triangle up to the middle row. For a given integer n, generate a pattern with 2n-1 rows. The first n rows should display the elements of Pascal's Triangle in increasing order, while the next n-1 rows should display them in decreasing order, forming a half-diamond.

Pascal’s Triangle is a triangular array of binomial coefficients. The value at position (i, j) in Pascal's Triangle is computed as C(i, j), where C(i, j) = i! / (j! * (i - j)!).

Example for n = 4:

Pattern Explanation:

• Row 1: C(0,0)

• Row 2: C(1,0) C(1,1)

• Row 3: C(2,0) C(2,1) C(2,2)

• Row 4: C(3,0) C(3,1) C(3,2) C(3,3)

• Row 5: Repeat Row 3

• Row 6: Repeat Row 2

• Row 7: Repeat Row 1

Test Cases:

Sample Input 1:

Input: n = 3

Sample Output 1:

Output:

1

1 1

1 2 1

1 1

1

Sample Input 2:

Input: n = 4

Sample Output 2:

Output:

1

1 1

1 2 1

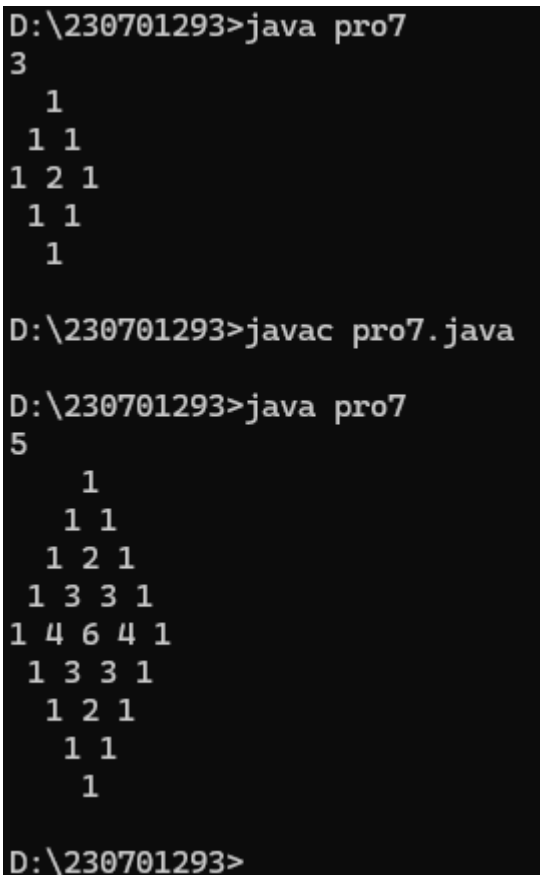
1 3 3 1

1 2 1

1 1

1

```
import java.util.Scanner;

public class pro7{
public static void main(String[] args) {
Scanner sc = new Scanner(System.in);
int n = sc.nextInt();
sc.close();

int[][] p = new int[n][];

for (int i = 0; i<n; i++) {
p[i] = new int[i + 1];
p[i][0] = 1;
p[i][i] = 1;
for (int j = 1; j<i; j++) {
p[i][j] = p[i - 1][j - 1] + p[i - 1][j];
}
}

for (int i = 0; i<n; i++) {
int spaces = (n - i - 1);

for (int j = 0; j<spaces; j++) {
System.out.print(" ");
}
for (int j = 0; j<= i; j++) {
System.out.print(p[i][j] + " ");
}
System.out.println();
}

for (int i = n - 2; i>= 0; i--) {
int spaces = (n - i - 1);

for (int j = 0; j < spaces; j++) {
System.out.print(" ");
}

for (int j = 0; j<= i; j++) {
System.out.print(p[i][j] + " ");
}
System.out.println();
}
}
}
```



8) We use the integers a, b, and n to create the following series:

(a+20

.b), (a+20

.b+21

.b),...,(a+20

.b+21

.b+.....+2n-1

.b)

You are given q queries in the form of a, b, and n. For each query, print the series

corresponding to the given a, b, and n values as a single line of n space-

separated integers.

Input Format

The first line contains an integer, q, denoting the number of queries.

Each line i of the q subsequent lines contains three space-separated integers describing the respective ai, bi, and ni values for that query.

Constraints

• 0 <= q <= 500

• 0 <= a,b <= 50

• 1 <= n <= 15

Output Format

For each query, print the corresponding series on a new line. Each series must be printed in order as a single line of n space-separated integers.

Sample Input

2

0 2 10

5 3 5

Sample Output

2 6 14 30 62 126 254 510 1022 2046

```java
import java.util.*;

public class pro8
{
public static void main(String args[])
{
Scanner sc = new Scanner(System.in);
int q = sc.nextInt();
while(q>0)
{

int a = sc.nextInt();
int b = sc.nextInt();
int n = sc.nextInt();
int d = 0;
while(d<=n)
{
int sum = a;
for(int i=0; i<d; i++)
sum += Math.pow(2, i)*b;
System.out.print(sum +" ");
d++;
}
q--;
}
}
}
```

8 14 26 50 98

```
D:\230701293>java pro8
2
0 2 10
0 2 6 14 30 62 126 254 510 1022 2046
4 9 16
4 13 31 67 139 283 571 1147 2299 4603 9211 18427
 36859 73723 147451 294907 589819
D:\230701293>
```