**NAME:SHAMRUDHA VARSHINI.P ROLL NO:231901048**

**QUESTION BASED ON DATATYPE (TYPE CONVERSION AND TYPE PROMOTION)**

**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&#39;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&#39;t be fitted anywhere.**

**-100000000000000 can be fitted in:**

**\* long**

**Solution:**

import java.util.Scanner;

public class BasicDataTypeFitter {

public static void main(String[] args) {

Scanner = new Scanner(System.in);

int t = scanner.nextInt();

for (int i = 0; i < t; i++) {

String input = scanner.next();

boolean canFit = false;

try {

long number = Long.parseLong(input);

System.out.print(input + " can be fitted in:\n");

if (number >= -128 && number <= 127) {

System.out.println("\* byte");

canFit = true;

}

if (number >= -32768 && number <= 32767) {

System.out.println("\* short");

canFit = true;

}

if (number >= -2147483648L && number <= 2147483647L) {

System.out.println("\* int");

canFit = true;

}

if (number >= -9223372036854775808L && number <= 9223372036854775807L) {

System.out.println("\* long");

canFit = true;

}

if (!canFit) {

System.out.println(input + " can't be fitted anywhere.");

}

} catch (NumberFormatException e) {

System.out.println(input + " can't be fitted anywhere.");

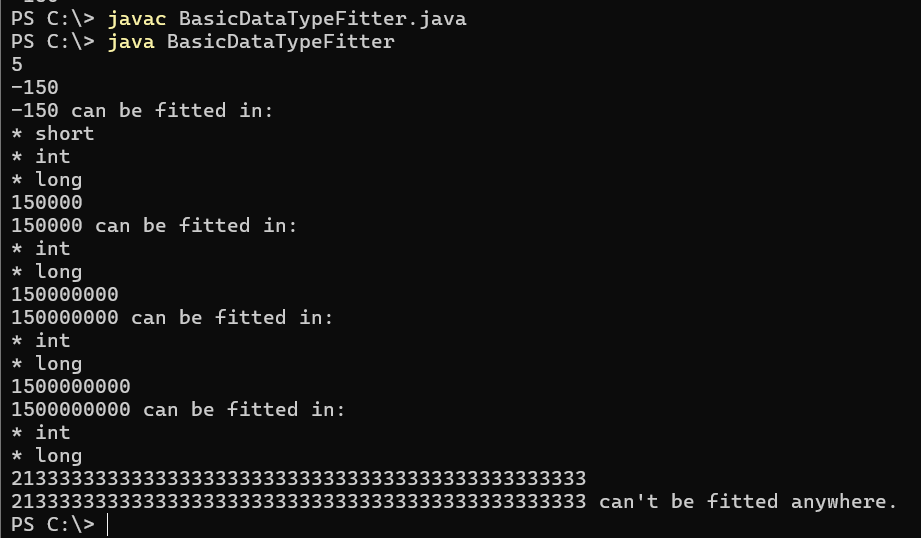
}

}

}

}

Output:



**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.

Solution:

import java.util.Scanner;

class Cent

{

public static void main(String[] args)

{

Scanner a=new Scanner(System.in);

int cent=a.nextInt();

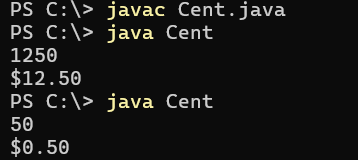
double b=cent/100.0;

System.out.printf("$" + "%.2f\n",b);

}

};

**Output:**



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).

**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.**

**Solution:**

import java.util.Scanner;

class Score

{

public static void main(String[] args)

{

Scanner o=new Scanner(System.in);

double a=o.nextDouble();

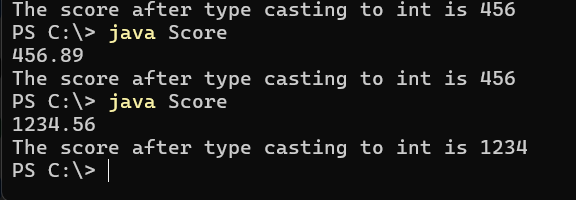
int b=(int)a;

System.out.println("The score after type casting to int is " +b);

}

};

**Output:**



**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).**

**Solution:**

import java.util.Scanner;

class Salary

{

public static void main(String[] args)

{

Scanner o=new Scanner(System.in);

int Csalary=o.nextInt();

double increasepercentage=o.nextDouble();

double formula=Csalary\*(increasepercentage/100.0);

double newsalary=formula+Csalary;

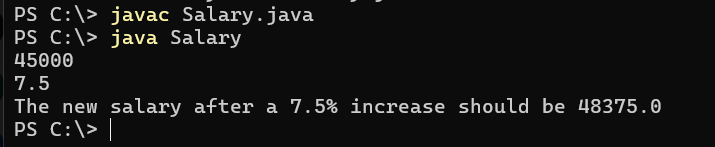
System.out.printf("The new salary after a %.1f%% increase should be %.1f\n",

increasepercentage,newsalary);

}

};

**Output:**



**Question - 1. 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.**

**Solution:**

import java.util.Scanner;

class Palindrome

{

public static void main(String[] args)

{

Scanner o=new Scanner(System.in);

int number=o.nextInt();

int remainder,reverse,temp;

temp=number;

reverse=0;

while(number>0)

{

remainder=number%10;

reverse=reverse\*10+remainder;

number/=10;

}

if(reverse==temp)

{

System.out.println("The reversed number is "+reverse+"."+" It is the same as the original.");

}

else

{

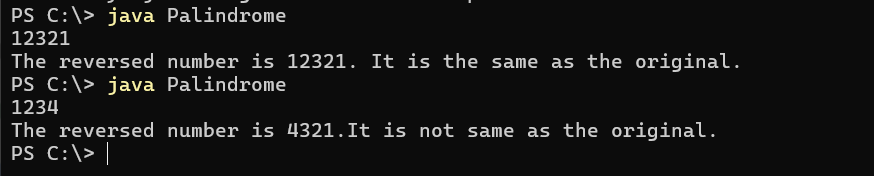
System.out.println("The reversed number is "+reverse+"."+"It is not same as the original.");

}

}

};

**Output:**



**Question - 2. 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:**

**\***

**\*\*\***

**\*\*\*\*\***

**\*\*\*\*\*\*\***

**\*\*\*\*\***

**\*\*\***

**\***

**Solution:**

import java.util.Scanner;

class Pattern

{

public static void main(String[] args)

{

int i, j, rows;

Scanner a=new Scanner(System.in);

rows=a.nextInt();

for (i = 1; i <= rows; i++) {

for (j = i; j < rows; j++) {

System.out.print(" ");

}

for (j = 1; j <= (2 \* i - 1); j++) {

System.out.print("\*");

}

System.out.println();

}

for (i = rows - 1; i >= 1; i--) {

for (j = rows; j > i; j--) {

System.out.print(" ");

}

for (j = 1; j <= (2 \* i - 1); j++) {

System.out.print("\*");

}

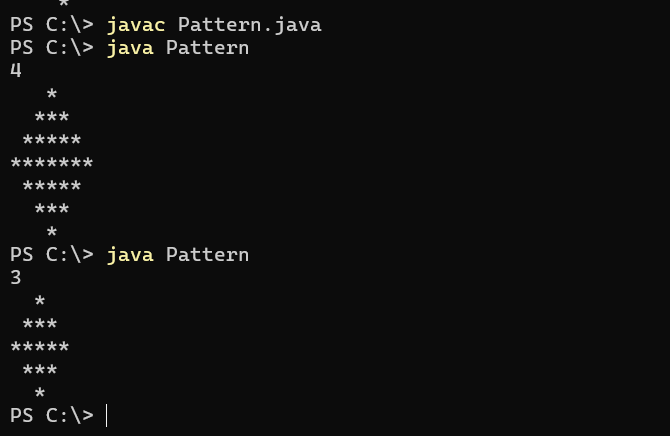
System.out.println();

}

}

};

**Output:**



**Question - 3. 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**

**Sample Input 3:**

**Input: n = 5**

**Sample Output 3:**

**Output:**

**1**

**1 1**

**1 2 1**

**1 3 3 1**

**1 4 6 4 1**

**1 3 3 1**

**1 2 1**

**1 1**

**1**

Explanation:

1. Pascal's Triangle Calculation:

o The triangle is built row by row, where each element is the binomial

coefficient calculated using factorials.

o For example, C(3,2) is calculated as 3! / (2! \* (3-2)!) = 3.

2. Pattern Construction:

o The first n rows display Pascal's Triangle in an expanding manner.

o The next n-1 rows reverse the pattern, forming a symmetric half-

diamond.

**Question - 4. 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**

**8 14 26 50 98**

**Explanation**

**We have two queries:**

**1. We use a=0, b=2, and n=10 to produce some series s0,s1,....sn-1:**

**o s0 = 0 + 1.2 = 2**

**o s1 = 0 + 1.2+2.2 = 6**

**o s2 = 0 + 1.2+2.2+4.2 = 14**

**... and so on.**

**Once we hit n=10, we print the first ten terms as a single line of space-separated**

**integers.**

**2. We use a=5, b=3, and n=5 to produce some series s0,s1,....sn-1:**

**o s0 = 5+1.3 = 8**

**o s1 = 5+1.3+2.3 = 14**

**o s2 = 5+1.3+2.3+4.3 = 26**

**o s3 = 5+1.3+2.3+4.3+8.3 = 50**

**o s4 = 5+1.3+2.3+4.3+8.3+16.3 = 98**

**We then print each element of our series as a single line of space-separated**

**values.**

**Solution:**

import java.util.Scanner;

public class SimpleSeriesGenerator {

public static void main(String[] args) {

Scanner = new Scanner(System.in);

int q = scanner.nextInt();

for (int i = 0; i < q; i++) {

int a = scanner.nextInt();

int b = scanner.nextInt();

int n = scanner.nextInt();

int[] series = new int[n];

for (int j = 0; j < n; j++) {

int currentValue = a;

int sum = 0;

int powerOfTwo = 1;

for (int k = 0; k <= j; k++) {

sum += powerOfTwo \* b;

powerOfTwo \*= 2;

}

series[j] = currentValue + sum;

}

for (int j = 0; j < n; j++) {

if (j > 0) {

System.out.print(" ");

}

System.out.print(series[j]);

}

System.out.println();

}

}

};

**Output:**

