You are given a class *Solution* and its *main* method in the editor. In each test cases, it takes an input  which represents a choice of the following:

* represents the volume of a cube that has to be calculated where  represents the length of the sides of the cube.
* represents the volume of a cuboid that has to be calculated where  represent the dimensions of a cuboid.
* represents the volume of a hemisphere that has to be calculated where  represents the radius of a hemisphere.
* represents the volume of a cylinder that has to be calculated where  represent the radius and height of the cylinder respectively.

Your task is to create the class *Calculate* and the required methods so that the code prints the volume of the figures rounded to exactly  decimal places.

In case any of the *dimensions* of the figures are , print *"java.lang.NumberFormatException: All the values must be positive"* without quotes and *terminate the program*.

*Note*: Use [Math.PI](http://docs.oracle.com/javase/7/docs/api/java/lang/Math.html" \l "PI) or  as the value of pi.

**Input Format**

First line contains , the number of test cases. Each test case contains *ch*, representing the choice as given in the problem statement.

* When *ch=1*, Next line contains , length of the sides of the cube.
* When *ch=2*, Next three lines contain , ,  representing length, breadth and height of the cuboid respectively. , ,  will be in three separate lines
* When *ch=3*, Next line contains , the radius of the hemisphere
* When *ch=4*, Next two lines contain ,  representing the radius and height of the cylinder respectively. ,  will be in two separate lines.

*Note:* You have to determine the *data type* of each parameter by looking at the code given in the *main* method.

**Constraints**   
   
   
There will be at most  digits after decimal point in input.

**Output Format**

For each test case, print the answer rounded up to exactly 3 decimal places in a single line. For example, *1.2345*should be rounded to *1.235*, *3.12995* should be rounded to *3.130*.

**Sample Input 1**

2

1

4

4

67.89

-98.54

**Sample Output 1**

64.000

java.lang.NumberFormatException: All the values must be positive

**Explanation**   
There are two test cases. In the first test case , means you have to calculate the volume of a cube. The next line contains the =4, means the side of the cube is . So the volume of the cube is .   
In the second test case, you have to calculate volume of a cylinder. But the height of the cylinder is negative, so an exception is thrown.

**Sample Input 2**

1

3

1.02

**Sample Output 2**

2.223

**import java.io.\*;**

**import java.util.\*;**

**import java.text.\*;**

**import java.math.\*;**

**import java.util.regex.\*;**

**import java.lang.reflect.\*;**

**import java.security.Permission;**

**public class Solution**

**{**

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

**Do\_Not\_Terminate.forbidExit();**

**try{**

**Calculate cal=new Calculate();**

**int T=cal.getINTVal();**

**while(T-->0){**

**double volume = 0.0d;**

**int ch=cal.getINTVal();**

**if(ch==1){**

**int a=cal.getINTVal();**

**volume=Calculate.get\_Vol().main(a);**

**}**

**else if(ch==2){**

**int l=cal.getINTVal();**

**int b=cal.getINTVal();**

**int h=cal.getINTVal();**

**volume=Calculate.get\_Vol().main(l,b,h);**

**}**

**else if(ch==3){**

**double r=cal.getDoubleVal();**

**volume=Calculate.get\_Vol().main(r);**

**}**

**else if(ch==4){**

**double r=cal.getDoubleVal();**

**double h=cal.getDoubleVal();**

**volume=Calculate.get\_Vol().main(r,h);**

**}**

**cal.output.display(volume);**

**}**

**}**

**catch (NumberFormatException e) {**

**System.out.print(e);**

**} catch (IOException e) {**

**e.printStackTrace();**

**}**

**catch (Do\_Not\_Terminate.ExitTrappedException e) {**

**System.out.println("Unsuccessful Termination!!");**

**}**

**}**

**}**

**/\*\***

**\*This class prevents the user from using System.exit(0)**

**\* from terminating the program abnormally.**

**\*/**

**class Do\_Not\_Terminate {**

**public static class ExitTrappedException extends SecurityException {**

**}**

**public static void forbidExit() {**

**final SecurityManager securityManager = new SecurityManager() {**

**@Override**

**public void checkPermission(Permission permission) {**

**if (permission.getName().contains("exitVM")) {**

**throw new ExitTrappedException();**

**}**

**}**

**};**

**System.setSecurityManager(securityManager);**

**}**

**}//end of Do\_Not\_Terminate**