```
/**
 * Angewandte Mathematik, SS11
 * Problem: 10338 - Mischievous Children
 * Link:
http://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=15&page=show_problem&problem=
1279
 *
 * Programm zum Ausrechnen der Anzahl der möglichen Aufreihungen von Wörtern
 */
import java.util.Scanner;

public class _1_Faculty {

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

		int cases = input.nextInt(); //Anzahl der Testwörter
		String	word; //eingegebene Wort
		int[] frequency; //Array zum speichern der Buchstaben in einem Array
		long number,denom,resultDenom; //zum Rechnen

		for(int c = 1; c <= cases;c++){
			word = input.next();
			frequency = new int[26];

			number = 1;
			resultDenom = 1;
			//Fakultät wird ausgerechnet und die Anzahl der Buchstaben im Array eins hochgezählt
			for(int i=word.length(); i>0;i--){
				number = number*i;

				frequency[word.charAt(i-1)-'A']++;
			}
			//Doppelte Buchstaben werden aufmultipliziert
			for(int i = 0; i < 26;i++){
				if(frequency[i] > 1) {
					denom = 1;
					for(int k=frequency[i]; k>1;k--){
						denom = denom*k;
					}
					resultDenom = resultDenom*denom;
				}
			}
			System.out.println("Data set "+c+": "+number/resultDenom);
		}
	}
}
```

```
/**
 * Angewandte Mathematik, SS11
 * Problem: 10213 - How Many Pieces Of Land?
 * Link:
http://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=14&page=show_problem&problem=
1154
 * Programm zum Berechnen der Anzahl der Lanstücke in einem Kreis mit n verbundenen Punkten am Rand
 */
import java.math.BigInteger;
import java.util.Scanner;
public class _2_Binomialkoeffizient {
	public static void main(String[] args) {

		Scanner input = new Scanner(System.in);

		int s; //Anzahl der Zeilen
		int n; //Anzahl der Punkte auf der Elipse
		s = input.nextInt();
		int fact4 = 24;
		int fact2 = 2;
		BigInteger numerator = BigInteger.ONE;
		BigInteger result;

		for(int c=0; c<s;c++){ //Abbruchbedingung, "s" Zeilen lesen
			result = BigInteger.ZERO;
			numerator = BigInteger.ONE;
			n = input.nextInt();
			if(n!=1 && n!=2 && n !=3 && n!=4){ //  Sonderfälle für n=1,2,3,4
				int nMinus4 = n-4;
				for(int i=n; i > nMinus4;i--){ // Mulitpliziere n*(n-1)*...*(n-4)
					numerator = numerator.multiply(BigInteger.valueOf(i));
				}

				result = numerator.divide(BigInteger.valueOf(fact4));// Dividiere das Produkt durch
24

				numerator = BigInteger.ONE;
				for(int i=n; i>(n-2);i--){ // Mulitpliziere n*(n-1)*(n-2)
					numerator = numerator.multiply(BigInteger.valueOf(i));
				}
				result = result.add(numerator.divide(BigInteger.valueOf(fact2)));
				//dividiere das Produkt durch 2 und addiere es auf

				result = result.add(BigInteger.ONE); // addiere 1, nach mosers kreisproblem
				System.out.println(result);
			}
			else{
				switch(n){
				case 1:
					System.out.println("1");
					break;
				case 2:
					System.out.println("2");
					break;
				case 3:
					System.out.println("4");
					break;
				case 4:
					System.out.println("8");
					break;
				}
			}
		}
	}
}
```

```
/**
 * Angewandte Mathematik, SS11
 * Problem: 485 - Pascal's Triangle of Death
 * Link:
http://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=6&page=show_problem&problem=4
26
 *
 * Programm zum Berechnen und Aufzeichnen eines linksjustierten Pascalschen Dreiecks
 * mit Hilfe der Addition der beiden Zahl die im Dreieck darüber stehen
 */

import java.math.BigInteger;

public class _3_1_PascalTriangle {

	public static void main(String[] args) {

		//Erstellt Endwert 10^60
		BigInteger finalValue = BigInteger.TEN;
		for(int c = 2; c <=10; c++) {
			finalValue = finalValue.multiply(BigInteger.TEN);
		}

		boolean end = false; //speichert ob der Endzustand einmal erreicht wurde
		int line = 1; //Speichert die Nr. und damit Elemente in einer Zeile

		//current-Array speichert die Werte der aktuellen Zeile
		BigInteger[] current = new BigInteger[9999];
		current[0] = current[1] = BigInteger.ONE;

		//next-Array speichert die Werte der nächsten Zeile, die gleichzeitig ausgegeben werden
		BigInteger[] next = new BigInteger[9999];

		//zeichnet die ersten beiden Elemente des Dreiecks auf
		System.out.println("1");
		System.out.println("1 1");

		//läuft bis der Endwert einmal erreicht/überschritten wurde
		while(!end){

			//erster Wert in jeder Zeile ist IMMER eins
			next[0] = BigInteger.ONE;

			//Werte "innerhalb" der Einser werden ausgerechnet
			for(int c = 1; c <= line; c++) {
				next[c] = current[c].add(current[c-1]);
				if(next[c].compareTo(finalValue) >= 0) end = true;
			}
			//Werte werde geprinted und die aktuell ausgegebene Zeile wird umgeschrieben
			for(int i = 0; i <= line; i++) {
				System.out.print(next[i]+" ");
				current[i] = next[i];
			}
			//letzte eins wird geprinted
			System.out.println("1");
			//es wird in die nächste Zeile gegangen und eine eins am Ende reingeschrieben
			line++;
			current[line] = BigInteger.ONE;
		}
	}
}
```

```
/**
 * Angewandte Mathematik, SS11
 * Problem: 10007 - Count the Trees
 * Link:
http://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=12&page=show_problem&problem=
948
 *
 * Programm zum Berechnen der Anzahl der möglichen Binärbäume mit n verschiedenen Elementen
 */

import java.math.BigInteger;
import java.util.Scanner;

public class _4_CatalanZahl {

	public static void main(String[] args) {

		Scanner input = new Scanner(System.in);
		int n,numerator,denumerator;
		BigInteger result = BigInteger.ONE; //Vorbelegung mit 1, neutrales Element bei Multiplikation
		n = input.nextInt();

		while(n!=0){
			/*
			 * CatalanZahl * N!
			 */
			numerator = 2*n;
			denumerator = n+2;

			for(int c = denumerator; c<=numerator; c++){ // entspricht dem Kürzen des Bruchs
				// dadurch wird nurnoch (2*n)*(2*(n-1))*...*(n+2) berechnet
				result = result.multiply(BigInteger.valueOf(c));
			}
			System.out.println(result);
			result = BigInteger.ONE; // Zurücksetzten für die nächste Eingabe
			n = input.nextInt(); // Einlesen der nächsten Zahl
		}
	}
}
```

```
/**
 * Angewandte Mathematik, SS11
 * Problem: 10007 - Count the Trees
 * Link:
http://uva.onlinejudge.org/index.php?option=com_onlinejudge&Itemid=8&category=12&page=show_problem&problem=
948
 *
 * Programm zum Berechnen der Möglichkeiten der Aufstellung von 2*1 DominoSteinen auf einem n*m großen Feld
 */
import java.math.*;
import java.util.Scanner;


public class _5_TilingDominoes {


	public static void main(String[] args) {
		Scanner input = new Scanner(System.in);
		int m;
		int n;

		double firstAddend; //Erster Teil der Summer
		double secondAddend; //zweiter teil der Summe
		double zw; //variable zum zwischenspeichern
		BigDecimal result = BigDecimal.ONE; //output



		while(input.hasNext()){
		m = input.nextInt();
		n = input.nextInt();
		if(m != 0 && ((m!=9)&&(n!=11))&&((m!=11)&&(n!=9))){ // sonderfall 9&11
			for(int j = 1; j <= m; j++){ //erste schleife für die "äußere" Multiplikation
				for(int k =1; k<=n; k++){ //zweite schleife für die "innere"
					firstAddend = ((Math.PI*j)/(m+1)); //erster Teil der Addition
					secondAddend = ((Math.PI*k)/(n+1)); //zweiter Teil der Addition
					zw = ((4*Math.pow(Math.cos(firstAddend),2))+
4*Math.pow(Math.cos(secondAddend),2));
					//Bilden der Summe
					result = (result.multiply(BigDecimal.valueOf((Math.pow(zw,0.25)))));
					//aufmultiplizieren mit vorherigem Wert und 4te Wurzel ziehen
				}
			}
		}
		else result =  BigDecimal.ZERO;
		result = result.divide(BigDecimal.ONE,0,4); //Division durch 1 mit korrekter Rundung
		System.out.println(result);
		result = BigDecimal.ONE;
		}
	}
}
```