# **Pattern Generator**

Write a program that repeatedly reads two numbers *n* and *k* and prints all bit patterns of length *n* with *k* ones in descending order (when the bit patterns are considered as binary numbers). You may assume that 30 >= *n* > 0, 8 > *k* >= 0, and *n* >= *k*. The first number in the input gives the number of pairs *n* and *k*. The numbers *n* and *k* are separated by a single space. Leading zeroes in a bit pattern should be included. See the example below.

## **Sample input**

3

2 1

2 0

4 2

## **Sample output**

The bit patterns are

10

01

The bit patterns are

00

The bit patterns are

1100

1010

1001

0110

0101

0011

| import java.io.\*;  public class PatternGenerator {  public static void patternGen(int index, int bits, String pattern)  {  if (index == 0 && bits == 0)  {  System.out.println(pattern);  }  else if (index != 0)  {  if (bits > 0)  {  patternGen(index - 1, bits -1, pattern+"1");  }  patternGen(index-1,bits,pattern+"0");  }  }    public static void main(String[] args)  {  try  {  String inputN;  int arr\_length;  int[][] arrayInt;  final int COLUMNS = 2;  BufferedReader br= new BufferedReader(new InputStreamReader(System.in));  inputN = br.readLine();  arrayInt = new int[Integer.parseInt(inputN)][COLUMNS];  arr\_length = Integer.parseInt(inputN);    for (int i = 0; i < arr\_length; i++)  {  inputN = br.readLine();  arrayInt[i][0] = Integer.parseInt(inputN.split(" ")[0]);  arrayInt[i][1] = Integer.parseInt(inputN.split(" ")[1]);    }    for (int i = 0; i < arr\_length; i++)  {  System.out.println("The bitwise pattern is: ");  patternGen(arrayInt[i][0], arrayInt[i][1], "");    }      } catch (IOException e) {}  }  } |
| --- |

Output:

|  |
| --- |