1. 문제제목

문제 1. 볼록다각형의 대각선 개수 구하기

1. 문제내용

‘볼록다각형’은 아래 설명대로 모든 내각이 180도 미만인 다각형을 말한다.

어느 날 중학교에서 ‘볼록다각형’과 ‘대각선’에 대해 배우고 온 귀여운 동생 유정이는 당신에게 “사각형은 대각선이 몇 개야~? 10각형은? 999각형은??” 하며 물어왔다. 볼록N각형의 대각선의 개수를 출력하는 프로그램을 작성하여 귀찮은 질문을 피해보자.

1. 테스트용 인풋, 아웃풋, 샘플용 3~4개

**INPUT FORMAT**

입력은 표준입력으로 제공된다.

입력은 여러 테스트 케이스가 포함될 수 있다.

첫째 줄 에는 테스트 케이스의 개수를 나타내는 자연수 T 가 주어진다. 이후 차례로 T개의 테스트 케이스가 주어진다 (1 ≤ T ≤ 100)

테스트 케이스마다 한 개의 N이 주어진다.

N은 3<N<1000의 범위의 자연수로 주어진다.

**OUTPUT FORMAT**

각 테스트 케이스의 답을 표준출력으로 출력해야 한다.

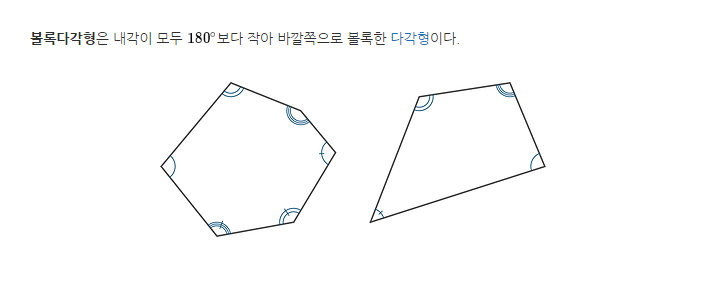
각 테스트 케이스마다 첫 줄에는 "Case #C"와 개행문자(newline) 출력하여야 한다. 이때 C는 테스트 케이스의 번호이다. 이후 압축된 결과와 개행문자(newline)를 출력한다.

|  |  |
| --- | --- |
| Input | Output |
| 4  4  5  6  7 | Case #1  2  Case #2  5  Case #3  9  Case #4  14 |

1. 소스

|  |
| --- |
| **Scanner** **sc** = **new** Scanner(**System**.***in***);    **int** **count** = sc.nextInt();    **int** **n** = 0;  **for**(**int** **i**=0;i<count;i++){  n = sc.nextInt();  **System**.***out***.println(n\*(n-3)/2);  } |

1. 그림



1. 문제제목

문제 2. 대칭수 판별하기

1. 문제내용

‘12321’과 같이 순서대로 읽은 수와 거꾸로 읽은 수가 일치하는 수를 대칭수 혹은 회문수라 한다. 여러 개의 긴 수가 입력될 때 각각의 수들이 대칭수인지 판별하는 프로그램을 작성하시오.

1. 테스트용 인풋, 아웃풋, 샘플용 3~4개

**INPUT FORMAT**

입력은 표준입력으로 제공된다.

입력은 여러 테스트 케이스가 포함될 수 있다.

첫째 줄 에는 테스트 케이스의 개수를 나타내는 자연수 T 가 주어진다. 이후 차례로 T개의 테스트 케이스가 주어진다 (1 ≤ T ≤ 100)

테스트 케이스마다 한 개의 정수가 주어진다

정수는 1~100 자리의 자연수 범위에서 주어진다.

**OUTPUT FORMAT**

각 테스트 케이스의 답을 표준출력으로 출력해야 한다.

각 테스트 케이스마다 첫 줄에는 "Case #C"와 개행문자(newline) 출력하여야 한다. 이때 C는 테스트 케이스의 번호이다. 이후 압축된 결과와 개행문자(newline)를 출력한다.

|  |  |
| --- | --- |
| Input | Output |
| 4  12345  1357531  99999999  115511 | Case #1  N  Case #2  Y  Case #3  Y  Case #4  Y |

1. 소스

|  |
| --- |
| Scanner sc = **new** Scanner(System.in);  **int** n;  String str;  String temp = "";  **while**(sc.hasNext()){  n=sc.nextInt();  str = Integer.*toString*(n);  **for**(**int** i=0;i<str.length();i++){  temp = temp + str.charAt(str.length()-i-1);  }  **if**(temp.equals(str)) System.out.println("Y");  **else** System.out.println("N");  temp = "";  } |

1. 문제제목

문제 3. 눈물의 학점포기

1. 문제내용

취준생 다현이는 저조한 평점을 받은 과목들에 대해 학점 포기를 하여 조금이라도 높은 평점을 이력서에 제출하려 한다. OO이가 재학중인 OO대학교의 학칙은 아래와 같다.

* 평균 평점 산출 방식은 (“각 과목평점 \* 각 학점”의 합)/(이수한 과목의 학점 수)
* 학점 포기는 1과목만 가능하다.
* 학점 포기 시 평균 평점은 (“포기 과목을 제외한 각 과목의 평점 \* 각 학점의 합”) / 이수한 과목의 학점 수 – 학점 포기 과목 학점 수) 이다.

학칙에 따라 평균 평점이 가장 높아지도록 학점 포기 과목을 선택하는 프로그램을 작성하시오.

1. 테스트용 인풋, 아웃풋, 샘플용 3~4개

**INPUT FORMAT**

입력은 표준입력으로 제공된다.

입력은 여러 테스트 케이스가 포함될 수 있다.

첫째 줄 에는 테스트 케이스의 개수를 나타내는 자연수 T 가 주어진다. 이후 차례로 T개의 테스트 케이스가 주어진다 (1 ≤ T ≤ 100)

테스트 케이스마다 과목명, 평점, 학점 수가 주어진다.

어떠한 과목을 포기하던 졸업 가능 여부에는 영향 없음,

평점은 4.5, 4.0, 3.5, 3.0, 2.5, 2.0 6가지 중 하나이며, 학점 수는 모든 과목이 1~4의 범위 내임,

모든 과목의 평점이 같아 학점포기를 해도 소용없는 경우는 발생하지 않음.

**OUTPUT FORMAT**

각 테스트 케이스의 답을 표준출력으로 출력해야 한다.

각 테스트 케이스마다 첫 줄에는 "Case #C"와 개행문자(newline) 출력하여야 한다. 이때 C는 테스트 케이스의 번호이다. 이후 압축된 결과와 개행문자(newline)를 출력한다.

|  |  |
| --- | --- |
| Input | Output |
| 1  5  미분적분학 3.5 3  C프로그래밍 3.0 3  객체지향프로그래밍 4.0 3  기초학술영어 2.0 2  데이터마이닝 3.0 2 | Case #1  기초학술영어 |

1. 소스

|  |
| --- |
| **public** **class** **test** {  **String** subjectName;  **double** pjum;  **int** hjum;    **public** **static** **int** **run**(**ArrayList**<test> subjects,**int** cnt){  **double** **max**=0;  **int** **hjumSum**=0;  **int** **index**=0;  **double** **pjumSum**=0;  **double** **meanPjum**=0;  **for**(**int** **i**=0;i<cnt;i++){  **for**(**int** **j**=0;j<cnt;j++){  **if**(j != i){  hjumSum += subjects.get(j).hjum;  pjumSum += subjects.get(j).pjum\*subjects.get(j).hjum;  }  }  meanPjum = pjumSum/hjumSum;  **if**(meanPjum > max) {  index = i;  max = meanPjum;  }  hjumSum = 0;  pjumSum = 0;  }  **return** index;  }  **public** **static** **void** **main**(**String** args[]){  **Scanner** **sc** = **new** Scanner(**System**.***in***);    **int** **count** = sc.nextInt();    **int** **n** = 0;    sc.nextLine();  **String** **sub**;  **int** **hscore**=0;  **double** **pscore**=0;    **ArrayList**<test> **Subjects** = **new** ArrayList<>();  **for**(**int** **i**=0;i<count;i++){  **test** **temp** = **new** test();  temp.subjectName=sc.next();  temp.pjum = sc.nextDouble();  temp.hjum = sc.nextInt();  sc.nextLine();  Subjects.add(temp);  }  **System**.***out***.println(Subjects.get(*run*(Subjects,count)).subjectName);  }  } |

1. 그림
2. 문제제목

문제 4. 정렬 검사

1. 문제내용

컴퓨터공학과 새내기 이린이는 프로그래밍 수업에서 여러가지 정렬 알고리즘에 대해 배웠다. 각 알고리즘을 사용하는 프로그램을 만들어 본 후 이번에는 주어진 수열이 정렬됐는지 검사하는 프로그램을 만들려 한다.

주어진 수열이 오름차순이면 “INC”, 내림차순이면 “DEC”를 출력하라. (정렬되지 않은 경우 NOTSORTED 출력)

1. 테스트용 인풋, 아웃풋, 샘플용 3~4개

**INPUT FORMAT**

입력은 표준입력으로 제공된다.

입력은 여러 테스트 케이스가 포함될 수 있다.

첫째 줄 에는 테스트 케이스의 개수를 나타내는 자연수 T 가 주어진다. 이후 차례로 T개의 테스트 케이스가 주어진다 (1 ≤ T ≤ 100)

테스트 케이스마다 한 개의 수열이 주어진다.

각 수열의 길이는 100 미만이다.

**OUTPUT FORMAT**

각 테스트 케이스의 답을 표준출력으로 출력해야 한다.

각 테스트 케이스마다 첫 줄에는 "Case #C"와 개행문자(newline) 출력하여야 한다. 이때 C는 테스트 케이스의 번호이다. 이후 압축된 결과와 개행문자(newline)를 출력한다.

|  |  |
| --- | --- |
| Input | Output |
| 4  1 2 3  3 2 1  1 3 5 6 7  1 5 3 8 9 | Case #1  INC  Case #2  DEC  Case #3  INC  Case #4  NOTSORTED |

1. 소스

|  |
| --- |
| **String**[] **str** = **new** **String**[100];  **String**[] **sortedStr** = **new** **String**[100];  **String** **buf** = "";  **for**(**int** **i**=0;i<count;i++){  buf = sc.nextLine();  str = buf.split(" ");  sortedStr = str.clone();  **Arrays**.*sort*(sortedStr);  **if**(**Arrays**.*equals*(str, sortedStr)) **System**.***out***.println("INC");  **else**{  sortedStr = str.clone();  **Arrays**.*sort*(sortedStr,**Collections**.*reverseOrder*());  **if**(**Arrays**.*equals*(str, sortedStr)) **System**.***out***.println("DEC");  **else** **System**.***out***.println("NOTSORTED");  }  } |

1. 문제제목

문제 5. 비밀번호 찾기

1. 문제내용

건망증이 심한 김깜빡씨는 그만 금고 비밀번호를 잊어버리고 말았다. 금고 자물쇠는 N자리의 자연수인 것만 기억하고 있으며 몇 차례 시도를 통해 틀린 비밀번호를 알고 있다. 김깜빡씨가 보다 빨리 비밀번호를 찾을 수 있도록 비밀번호일 가능성이 있는(이미 시도한 비밀번호를 제외한) 모든 경우의 비밀번호를 출력하라.

1. 테스트용 인풋, 아웃풋, 샘플용 3~4개

**INPUT FORMAT**

입력은 표준입력으로 제공된다.

입력은 여러 테스트 케이스가 포함될 수 있다.

첫째 줄 에는 테스트 케이스의 개수를 나타내는 자연수 T 가 주어진다. 이후 차례로 T개의 테스트 케이스가 주어진다 (1 ≤ T ≤ 100)

테스트 케이스의 첫 줄에는 틀린 것으로 검증 된 비밀번호의 수와 비밀번호 수의 자리수가 입력된다.

이후 틀린 것으로 검증 된 비밀번호들이 각 줄에 하나씩 입력된다.

**OUTPUT FORMAT**

각 테스트 케이스의 답을 표준출력으로 출력해야 한다.

출력순서는 오름차순이다.

각 테스트 케이스마다 첫 줄에는 "Case #C"와 개행문자(newline) 출력하여야 한다. 이때 C는 테스트 케이스의 번호이다. 이후 압축된 결과와 개행문자(newline)를 출력한다.

|  |  |
| --- | --- |
| Input | Output |
| 1  5 2  00  03  05  09  88 | Case #1  01  02  04  06  07  08  10  .. (11~86)  87  89  .. (90~98)  99 |

1. 소스

|  |
| --- |
| **public** **static** **void** **main**(**String** args[]){  **Scanner** **sc** = **new** Scanner(**System**.***in***);    **int** **count** = sc.nextInt();  **int** **len** = sc.nextInt();    sc.nextLine();  **String** **option** = "%0" + len + "d";  **ArrayList**<String> **exception** = **new** ArrayList<>();  **int** **idx** = 0;  **for**(**int** **i**=0;i<count;i++){  exception.add(sc.nextLine());  }  **for**(**int** **i**=0;i<**Math**.*pow*(10,len);i++){  **String** **temp** = **String**.*format*(option, i);  **if**(!exception.contains(temp))  **System**.***out***.println(temp);  }  } |

1. 그림
2. 문제제목

문제 6. 까다로운 애인

1. 문제내용

얼마 전 모태솔로에서 탈출한 김계획씨의 애인은 황당한 통보를 해왔다. 그 내용은 3주 동안 5번 미만의 만남을 가질 경우 삐지겠다는 것이었다. 평소 모든 일을 계획에 따라 행동하기 좋아하는 김계획씨는 N주 동안의 만남 횟수를 미리 계획하기로 했다. 계획씨의 애인이 통보해온 경우가 아니라면 절대 삐지지 않고 만남의 횟수가 계획대로 반드시 진행된다 가정할 때 N주 동안 김계획씨의 애인이 몇 차례나 삐지게 될 지 예상하는 프로그램을 작성하라.

(단 계획의 시작부터 둘째 주 까지는 김계획씨의 애인은 삐지지 않으며 삐짐은 그 다음주로 넘어가기 전에 무조건 풀림.)

1. 테스트용 인풋, 아웃풋, 샘플용 3~4개

**INPUT FORMAT**

입력은 표준입력으로 제공된다.

입력은 여러 테스트 케이스가 포함될 수 있다.

첫째 줄 에는 테스트 케이스의 개수를 나타내는 자연수 T 가 주어진다. 이후 차례로 T개의 테스트 케이스가 주어진다 (1 ≤ T ≤ 100)

각 케이스의 첫 번째 줄에는 몇 주간의 계획인지 입력된다.

이후 각 라인은 각 주마다의 만남 횟수가 입력된다.

**OUTPUT FORMAT**

각 테스트 케이스의 답을 표준출력으로 출력해야 한다.

각 테스트 케이스마다 첫 줄에는 "Case #C"와 개행문자(newline) 출력하여야 한다. 이때 C는 테스트 케이스의 번호이다. 이후 압축된 결과와 개행문자(newline)를 출력한다.

|  |  |
| --- | --- |
| Input | Output |
| 1  10  1  2  2  1  0  3  2  1  2  2 | Case #1  2 |

1. 소스

|  |
| --- |
| **public** **static** **void** **main**(**String** args[]){  **Scanner** **sc** = **new** Scanner(**System**.***in***);  **int** **count** = sc.nextInt();  sc.nextLine();  **int** **prediction** = 0;  **ArrayList**<Integer> **date** = **new** ArrayList<>();  **for**(**int** **i**=0;i<count;i++){  date.add(sc.nextInt());  }  **for**(**int** **i**=2;i<count;i++){  **if**(date.get(i-2)+date.get(i-1)+date.get(i) < 5){  prediction++;  }  }  **System**.***out***.println(prediction);  } |

1. 그림
2. 문제제목

문제 7. 좌표 산출

1. 문제내용

뼛속까지 공대생인 정환이는 어느 날 두더지게임을 하고 두더지들에게 잔뜩 약이 올라 두더지게임 오락기의 영상을 이진 데이터로 처리하여 자동으로 두더지들의 머리를 내려치는 기계를 개발하려 한다. 영상처리를 통해 얻은 이진데이터를 망치 기계에 좌표를 전달하려 할 때, 0 사이에 있는 1의 좌표를 출력하라.

(3행 4열에 위치한 경우 (3,4) 출력)

1. 테스트용 인풋, 아웃풋, 샘플용 3~4개

**INPUT FORMAT**

입력은 표준입력으로 제공된다.

입력은 여러 테스트 케이스가 포함될 수 있다.

첫째 줄 에는 테스트 케이스의 개수를 나타내는 자연수 T 가 주어진다. 이후 차례로 T개의 테스트 케이스가 주어진다 (1 ≤ T ≤ 100)

테스트 케이스의 첫 줄에는 두더지게임 오락기의 크기가 입력된다 (n\*n)

**OUTPUT FORMAT**

각 테스트 케이스의 답을 표준출력으로 출력해야 한다.

각 테스트 케이스마다 첫 줄에는 "Case #C"와 개행문자(newline) 출력하여야 한다. 이때 C는 테스트 케이스의 번호이다. 이후 압축된 결과와 개행문자(newline)를 출력한다.

출력 순서는 먼저 위에서 아래 순, 같은 행에서는 왼쪽에서 오른쪽 순으로 출력한다.

|  |  |
| --- | --- |
| Input | Output |
| 1  5  00000  00000  00010  00000  01001 | Case #1  (3,4)  (5,2)  (5,5) |

1. 소스

|  |
| --- |
| **public** **static** **void** **main**(**String** args[]){  **Scanner** **sc** = **new** Scanner(**System**.***in***);    **int** **count** = sc.nextInt();    sc.nextLine();    **String** **empty** = "";  **String** **buf** = "";  **for**(**int** **i**=0;i<count;i++){  empty = empty + "0";  }  **for**(**int** **i**=0;i<count;i++){  buf = sc.nextLine();  **if**(!empty.equals(buf)){  **for**(**int** **j**=0;j<count;j++){  **if**(buf.charAt(j) == '1')  **System**.***out***.println("("+(i+1)+","+(j+1)+")");  }  }    }    } |

1. 그림
2. 문제제목

문제 8. HANWHA를 찾아라

1. 문제내용

임의의 문자열 안에 H,A,N,W,H,A 알파벳이 순서대로 배치되어 있는지 확인하고 순서대로 배치되어 있는 경우 HANWHA. 만약 순서대로 배치되어 있지 않은 경우 NOT HANWHA 를 출력하라.

(XHXANWXHA,HHANWHA,AHANWHA는 순서대로 배치된 문자열이지만 HAANWHA, XHXANHWA 는 순서대로 배치되어 있는 문자열이 아니다.)

(순서대로 배치 되어 있는 경우 시작점과 끝점을 출력하게 하면 난이도 상승)

1. 테스트용 인풋, 아웃풋, 샘플용 3~4개

**INPUT FORMAT**

입력은 표준입력으로 제공된다.

입력은 여러 테스트 케이스가 포함될 수 있다.

첫째 줄 에는 테스트 케이스의 개수를 나타내는 자연수 T 가 주어진다. 이후 차례로 T개의 테스트 케이스가 주어진다 (1 ≤ T ≤ 100)

테스트 케이스마다 한 개의 문자열이 주어진다.

**OUTPUT FORMAT**

각 테스트 케이스의 답을 표준출력으로 출력해야 한다.

각 테스트 케이스마다 첫 줄에는 "Case #C"와 개행문자(newline) 출력하여야 한다. 이때 C는 테스트 케이스의 번호이다. 이후 압축된 결과와 개행문자(newline)를 출력한다.

|  |  |
| --- | --- |
| Input | Output |
| 5  XXXHXAXNXWHXXAX  HANWHA  ANWHA  ANWHANWHA  ZCHXATNIOWPHTATH | Case #1  HANWHA  Case #2  HANWHA  Case #3  NOT HANWHA  Case #4  HANWHA  Case #5  HANWHA |

1. 소스

|  |
| --- |
| **public** **static** **void** **main**(**String** args[]){  **Scanner** **sc** = **new** Scanner(**System**.***in***);    **int** **count** = sc.nextInt();  sc.nextLine();  **String** **buffer** = "";  **String** **check** = "";  **for**(**int** **i**=0;i<count;i++){  buffer = sc.nextLine();  **for**(**int** **j**=0;j<buffer.length();j++){  **if**(buffer.charAt(j) == 'H' || buffer.charAt(j) == 'A' || buffer.charAt(j) == 'N' || buffer.charAt(j) == 'W'){  check = check + buffer.charAt(j);  }  }  **if**(check.contains("HANWHA")){  **System**.***out***.println("HANWHA");  }  **else**  **System**.***out***.println("NOT HANWHA");  check = "";  }  } |

1. 그림
2. 문제제목

문제 9. 피보나치 나선

1. 문제내용

피보나치 수열이란 0과 1로 시작하여 다음 피보나치 수는 바로 앞의 두 피보나치 수의 합이 된되는 수열을 말한다. 피보나치 나선은 아래 그림과 같이 안쪽에서부터 한 변의 길이가 피보나치 수열에 해당하는 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89... 인 정사각형마다 사분원을 그려서 나선 모양이 되도록 이은 것을 말한다. 이 과정에서 그려지는 n번째 사각형의 넓이를 구하라

(테스트 케이스를 다수로 하고 제한시간을 두어 DP 알고리즘 사용 유도)

1. 테스트용 인풋, 아웃풋, 샘플용 3~4개

**INPUT FORMAT**

입력은 표준입력으로 제공된다.

입력은 여러 테스트 케이스가 포함될 수 있다.

첫째 줄 에는 테스트 케이스의 개수를 나타내는 자연수 T 가 주어진다. 이후 차례로 T개의 테스트 케이스가 주어진다 (1 ≤ T ≤ 100)

테스트 케이스마다 한 개의 N이 주어진다

**OUTPUT FORMAT**

각 테스트 케이스의 답을 표준출력으로 출력해야 한다.

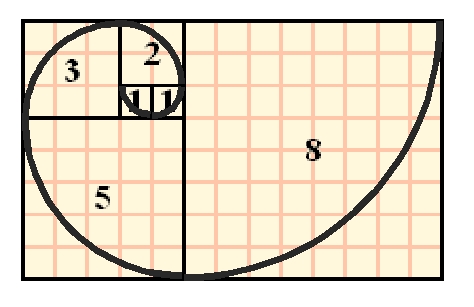
각 테스트 케이스마다 첫 줄에는 "Case #C"와 개행문자(newline) 출력하여야 한다. 이때 C는 테스트 케이스의 번호이다. 이후 압축된 결과와 개행문자(newline)를 출력한다.

|  |  |
| --- | --- |
| Input | Output |
| 3  1  3  5 | Case #1  1  Case #2  4  Case #3  25 |

1. 소스

|  |
| --- |
| **public** **static** **void** **main**(**String** args[]){  **Scanner** **sc** = **new** Scanner(**System**.***in***);    **int** **count** = sc.nextInt();  **int** **max**=2;  **ArrayList**<Long> **fib** = **new** ArrayList<>();  fib.add((**long**) 1);  fib.add((**long**) 1);  sc.nextLine();    **int** **temp**=0;  **for**(**int** **i**=0;i<count;i++){  temp = sc.nextInt();  **if**(temp>max){  **for**(**int** **j** = fib.size();j<temp;j++){  fib.add(fib.get(j-1)+fib.get(j-2));  }  }  **System**.***out***.println((fib.get(temp-1)\*fib.get(temp-1)));  sc.nextLine();  }  } |

1. 그림



1. 문제제목

문제 10. 알고리즘 찾기

1. 문제내용

주어진 수를 이진수로 변환한 후 N개의 0을 가장 낮은 자리에 추가한 값을 십진수로 출력하시오

(알고리즘을 찾는다면 주어진 수에 2^N을 곱하는 프로그램 작성, 못 찾는다면 시간 소모)

1. 테스트용 인풋, 아웃풋, 샘플용 3~4개

**INPUT FORMAT**

입력은 표준입력으로 제공된다.

입력은 여러 테스트 케이스가 포함될 수 있다.

첫째 줄 에는 테스트 케이스의 개수를 나타내는 자연수 T 가 주어진다. 이후 차례로 T개의 테스트 케이스가 주어진다 (1 ≤ T ≤ 100)

테스트 케이스마다 주어진 수와 추가될 0의 개수가 입력된다.

**OUTPUT FORMAT**

각 테스트 케이스의 답을 표준출력으로 출력해야 한다.

각 테스트 케이스마다 첫 줄에는 "Case #C"와 개행문자(newline) 출력하여야 한다. 이때 C는 테스트 케이스의 번호이다. 이후 압축된 결과와 개행문자(newline)를 출력한다.

|  |  |
| --- | --- |
| Input | Output |
| 3  25 1  38 2  40 3 | Case #1  50  Case #2  152  Case #3  320 |

1. 소스

|  |
| --- |
| **public** **static** **void** **main**(**String** args[]){  **Scanner** **sc** = **new** Scanner(**System**.***in***);    **int** **count** = sc.nextInt();    sc.nextLine();    **int** **base** = 0;  **int** **exp** = 0;    **for**(**int** **i**=0;i<count;i++){  base = sc.nextInt();  exp = sc.nextInt();  **System**.***out***.println((**int**)(base\***Math**.*pow*(2, exp)));  sc.nextLine();  }  }  } |

1. 그림