|  |
| --- |
| Codescape Consultants Pvt. Ltd. |
| Personal Interview Problem Set (Programming) |
| 2012-13 |

**Q1** *[SCORE = 30]*. Without Using any CONDITIONAL STATEMENT [i.e if-else, switch] print the first EVEN DIGIT from the RIGHT, if exists else print -1.

**Input (Q1.in)**

File **Q1.in** first line will have an Integer value [x], where 0 < x < 100

From Next line till the EOF will contain an integer number [y], where 0 < y < 1000000, one integer (i.e one test case input) per line.

**Output (Q1.out)**

File should be **Q1.out;** it will have first case as number of test cases [x]

And then one output per line, represented as

Case#<output number> = <Output>

**Sample Input**

4

123

123456

654321

3

**Sample Output**

4

Case #1 = 2

Case#2 = 6

Case#3 = 2

Case#4 = -1

**Q2** *[SCORE = 70]*. Write a program to convert positive integer, represented in ENGLISH LANGUAGE format *( eg. 1234 is integer represented in ENGLISH LANGUAGE is ONE THOUSAND TWO HUNDRED THIRTY FOUR)*, into Roman numbers.

Assume that the numbers to be converted is less than 4000.

The rule for constructing a Roman number is assumed to be as follows-

* In Roman number system, **i** is the symbol for 1, **v** for 5, **x** for 10, **l** for 50, **c** for 100, **d** for 500 and **m** for 1000.
* Symbols with larger values usually appear before symbols with smaller values. The value of a Roman number is, in general, the sum of the values of the symbols. For example, **ii** is 2, **viii** is 8.
* However, if a symbol with smaller value appears before a symbol with larger value, the value of these two symbols is the difference of the two values. For example, **iv** is 4, **ix** is 9, and **lix** is 59.
* Note that no four consecutive symbols in the Roman number can be the same. For example, **iv**, but not **iiii**, is the Roman number 4.
* Never use **vv** for 10, **ll** for 100, **dd** for 1000, or **vvv** for 15, etc.

The Roman numbers constructed in this way may not be unique. For example, both **mcmxc** and **mxm** are valid for 1990; the roman number generated by your program need not be the shortest one.

**Input (Q2.in)**

File **Q2.in** first line will have an Integer value [x], where 0 < x < 100

From Next line till the EOF will contain an integer number [y], where 0 < y < 4000, one number in ENGLISH LANGUAGE FORMAT (i.e one test case input) per line.

**Output (Q2.out)**

File should be **Q2.out;** it will have first case as number of test cases [x]

And then one output per line, represented as

Case#<output number> = <Output>

**Sample Input**

6

Three

Eight

One Hundred Seventy Two

Four

Nineteen Hundred Ninety Nine

Five

**Sample Output**

6

Case#1 = iii

Case#2 = viii

Case#3 = clxxii

Case#4 = iv

Case#5 = mcmxc

Case#6 = v

**Q3** *[SCORE = 30]*. Summation of sequence of integers is always a common problem in Computer Science. Rather than computing blindly, some intelligent techniques make the task simpler. Here you have to find the summation of a sequence of integers. The sequence is an interesting one and it is the all possible permutations of a given set of digits. For example, if the digits are <1 2 3>, then six possible permutations are <123>, <132>, <213>, <231>, <312>, <321> and the sum of them is 1332.

**Input (Q3.in)**

First line will be number of test cases.

Each input set will start with a positive integer N (1≤N≤9).

The next line will contain N decimal digits. Input will be terminated by N=0. There will be at most 2000 test set.

**Output (Q3.out)**

First line will be number of test cases.

For each test set, there should be a one line output containing the summation. The value will fit in 64-bit unsigned integer.

**Sample Input**

2

3

1 2 3

3

1 1 2

**Sample Output**

2

Case#1 = 1332

Case#2 = 444

*[Score 70]* **Q4**. Delete the least number of integers from a given set of integers so that the product of the remaining integers in the set is a perfect square. In case there is more than one solution then find the solution that gives the largest perfect square. Assume that each integer contains five or less number of digits. The total number of integers in the given set is twenty or less.

**Input (Q4.in)**

First line will be number of test cases

The input may contain multiple test cases.

For each test case there is a single input line. The line contains the given set of integers. The

**Output (Q4.out)**

For each test case there is only one output line. The line simply prints the integers to be deleted

in ascending order. There are two special cases; print output for these cases as indicated below.

Case 1: No integer is to be deleted: Print 0 as output.

Case 2: All integers are to be deleted: Print all integers in ascending order.

**Sample Input**

4

2 3 12 18 24

12 10 15 18

4 12 10 15

10 12 15

**Sample Output**

4

Case#1 = 24

Case#2 = 0

Case#3 = 10 12 15

Case#4 = 10 12 15

**Q5.** Farmer John has built a new long barn, with N (2 <= N <= 100,000) stalls. The stalls are located along a straight line at positions x1,...,xN (0 <= xi <= 1,000,000,000).

His C (2 <= C <= N) cows don't like this barn layout and become aggressive towards each other once put into a stall. To prevent the cows from hurting each other, FJ want to assign the cows to the stalls, such that the minimum distance between any two of them is as large as possible. What is the largest minimum distance?

**Input (Q5. in)**

t – the number of test cases, then t test cases follows.

\* Line 1: Two space-separated integers: N and C

\* Lines 2..N+1: Line i+1 contains an integer stall location, xi

**Output (Q5. Out)**

For each test case output one integer: the largest minimum distance.

**Example**

**Input:**

1

5 3

1

2

8

4

9

**Output:**

3

**Output details:**

FJ can put his 3 cows in the stalls at positions 1, 4 and 8,

Resulting in a minimum distance of 3.