# Rules

Competition rules are based on those used at the ACM Regional programming competitions sponsored by the ACM (Association for Computing Machinery), though there are some differences because the student level and experience are not the same.

- **Every high school in Ontario is eligible to send a team.**
- **Normally two teams per participating high school. If a high school wishes to bring additional teams for the experience, they may be registered and participate as "spares" if space permits. The "spares" do not receive any prizes or ranking scores in the final standings.**
- **There are generally three students per team though a team of two is also allowed (but at a disadvantage!).**
- **There is only one computer workstation per team, and only one login session per team is permitted.**
- **Students may use hard-copy (paper or book) reference material but not soft-copy (electronic) reference material. So no disks or CDs are permitted, not even Internet.**
- **No calculators, audio devices or video devices are permitted.**
- **New: This year we are using the official contest environment used by the ACM Regional Programming Competition called PC^2.**
- **No communication is permitted between teams or between teams and teachers/coaches, once the competition has started. Also, students are not permitted access to outside resources via a web browser or e-mail during the competition.**
- **Solutions are submitted electronically as instructed at the time of the competition, and to be accepted a submitted program must produce the right output values in the correct format, for each of the sample input data files. Program code itself is not read and not evaluated in any way. Input test data files used by the judges may (and generally will) include one or more data files that the teams have not seen.**
- **A solution which is not accepted as 'correct' will be rejected. A time penalty of 20 minutes is assessed for each rejected submission.**

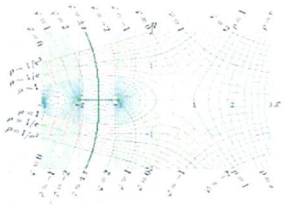
**The winner is determined by most problems solved, with ties broken by total time taken.**

# A

# Last Factorial Digit

The factorial of $N$, written as $N!$, is defined as the product of all the integers from 1 to $N$. For example, $3! = 1 \times 2 \times 3 = 6$.

This number can be very large, so instead of computing the entire product, just compute the last digit of $N!$ (when $N!$ is written in base 10).



*Factorials on the complex plane, by Dmitrii Kouznetsov*

## Input

The first line of input contains a positive integer $1 \leq T \leq 10$, the number of test cases. Each of the next $T$ lines contains a single positive integer $N$. $N$ is at most 10.

## Output

For each value of $N$, print the last digit of $N!$.

### Sample Input 1

```
3
1
2
3
```

### Sample Output 1

```
1
2
6
```

### Sample Input 2

```
2
5
2
```

### Sample Output 2

```
0
2
```

# B

# Coffee Cup Combo

Jonna is a university student who attends $n$ lectures every day. Since most lectures are way too simple for an algorithmic expert such as Jonna, she can only stay awake during a lecture if she is drinking coffee. During a single lecture she needs to drink exactly one cup of coffee to stay awake.

Some of the lecture halls have coffee machines, so Jonna can always make sure to get coffee there. Furthermore, when Jonna leaves a lecture hall, she can bring at most two coffee cups with her to the following lectures (one cup in each hand). But note that she cannot bring more than two coffee cups with her at any given time.

Given which of Jonna's lectures have coffee machines, compute the maximum number of lectures during which Jonna can stay awake.

## Input

The first line contains the integers $n$ ($1 \leq n \leq 10^5$), the number of lectures Jonna attends.

The second line contains a string $s$ of length $n$. The $i$'th letter is `1` if there is a coffee machine in the $i$'th lecture hall, and otherwise it is `0`.

## Output

Print one integer, the maximum number of lectures during which Jonna can stay awake.

### Sample Input 1

```
10
0100010100
```

### Sample Output 1

```
8
```

## Sample Input 2

```
10
1100000000
```

## Sample Output 2

```
4
```

## Sample Input 3

```
1
0
```

## Sample Output 3

```
0
```

# C

# Square Peg in a Round Hole

Mr. Johnson likes to build houses. In fact, he likes it so much that he has built a lot of houses that he has not yet placed on plots. He has recently acquired $N$ circular plots. The city government has decided that there can be only one house on each plot, and a house cannot touch the boundary of the plot.

Mr. Johnson has $M$ circular houses and $K$ square houses. Help him figure out how many of the plots he can fill with houses so that he can get some money back on his investments.

## Input

The first line of input consists of 3 space-separated integers $N$, $M$, and $K$. The second line contains $N$ space-separated integers, where the $i^{\text{th}}$ integer denotes the radius $r_i$ of the $i^{\text{th}}$ plot. The third line contains $M$ space-separated integers, where the $i^{\text{th}}$ integer denotes the radius $r_i$ of the $i^{\text{th}}$ circular house. The fourth line contains $K$ space-separated integers, where the $i^{\text{th}}$ integer denotes the side length $s_i$ of the $i^{\text{th}}$ square house.

## Output

Output the largest number of plots he can fill with houses.

## Limits

- $1 \leq N, M, K, r_i, s_i \leq 100$

### Sample Input 1

```
5 3 3
1 2 6 7 8
2 6 7
4 8 9
```

### Sample Output 1

```
3
```

# D

# Bus Numbers

A famous story about the mathematicians *G.H. Hardy* and *Srinivasa Ramanujan* goes as follows (as told by Hardy):

> *I remember once going to see him (Ramanujan) when he was lying ill at Putney. I had ridden in taxicab No. 1729, and remarked that the number seemed to be rather a dull one, and that I hoped it was not an unfavourable omen. "No", he replied, "it is a very interesting number; it is the smallest number expressible as the sum of two [positive] cubes in two different ways."*


*A bus with a number that is not a bus number according to our definition. License: public domain*

It is from this story the *taxicab numbers* got their name. The $n$'th taxicab numbers is defined to be the smallest number that can be expressed as a sum of two *positive* cube numbers in $n$ distinct ways.

It turns out that these numbers grows rather quickly. This makes them very hard to compute, which is not very fun. A variation of the concept is to consider what we will call the *bus numbers* – all the numbers which can expressed as the sum of two *positive* cube numbers in *at least* 2 distinct ways. Note that according to this definition, all taxicab numbers (except the first) are also bus numbers.

Your task is to write a program that generates bus numbers; in particular, the *largest* bus number that is *at most* equal to some limit $m$.

## Input

The input consists of:

- one line with an integer $m$ ($1 \leq m \leq 400\,000$), the upper bound of the bus number.

## Output

Output the largest bus number $x$ which does not exceed $m$. If there is no such number, output none.

### Sample Input 1

```
1730
```

### Sample Output 1

```
1729
```

### Sample Input 2

```
100
```

### Sample Output 2

```
none
```

# E

# Chocolate Division

Alf and Beata were two young adults living together a long, long time ago, before you could spend all your afternoons competing in programming. Their lives were thus much more boring than those of today's young adults. How could you even survive back then, you might ask yourself. The answer is simple: you make your own chocolate bars! Our two cohabitants loved making chocolate bars, and often had huge piles of them each afternoon. To avoid filling their entire living room with chocolate bars, Beata challenged her friend to a game every evening to eat up all the chocolate – the Chocolate Division game.

The Chocolate Division game is played by two players (in our case, Alf and Beata) using a chocolate bar consisting of $R$ rows, each containing $C$ squares of chocolate. A move consists of splitting a rectangular piece of chocolate either horizontally or vertically into two pieces, such that the two new pieces again have integer height and width. The first move is performed on the original chocolate bar. Each subsequent move can performed on any piece of chocolate that has been split off so far, as long as it doesn't have dimensions $1 \times 1$ (no split can be done on such a piece). Alf starts with the first move. If, at any point, a person can not perform their move because all remaining pieces have size $1 \times 1$ squares, that person loses.

Can you compute who wins the game, if both players play optimally?

## Input

The first and only line contains the integers $R$ and $C$ ($1 \leq R, C \leq 500$), the height and width of the original chocolate bar.

## Output

Output `Alf` if Alf wins the game, or `Beata` if Beata wins the game.

### Sample Input 1

```
1 1
```

### Sample Output 1

```
Beata
```

### Sample Input 2

```
1 2
```

### Sample Output 2

```
Alf
```

### Sample Input 3

```
2 2
```

### Sample Output 3

```
Alf
```