Exercise – *Defensive Line*

Believe it or not, American football has become an increasingly tactical game. You just got hired as an algorithmic tactics assistant for the AlgoLand Provers. They are in a deep crisis since their game strategies—although provably optimal—turned out to be inefficient in real world scenarios. As a result, the team's archrivals, the HeuristicCity Mighty Bugs, have been winning the championship for several years. It is time to make the Provers great again and your first task is to revise the offensive strategy.

The scenario is as follows. At the start of a football game, the attacking team and the defending team line up and face each other. The n defenders form a 'wall' of players (called the defensive line) from the left side (0-th defender) of the field to the right side ($(n-1)$-st defender). The m attackers then try to break through this wall and advance the football down the field, with the ultimate goal of scoring a touchdown. However, there are certain constraints on how attackers can do this depending on their skills, physical and mental strength, and the rules of the game.

Each defender has a certain *defense value* and each attacker has a certain *attack strength*. Let v_i denote the defense value of the i-th defender. For simplicity, you may assume that all attackers have the same attack strength k. Every attacker must attack a nonempty segment of defenders standing next to each other. In addition, the attack strength must be at least as large as the sum of the defense values of the defenders within the segment. However, if the attack strength is strictly larger than this sum, then the attack is considered too violent and counts as a *foul*. It also counts as a foul if a defender is attacked by two or more attackers.

Your task is to derive an offensive strategy and tell your attackers where to attack such that the total number of attacked defenders is maximised and no foul is committed. Formally, a *legal strategy* is a choice of disjoint intervals $[a_0, b_0], \ldots, [a_{m-1}, b_{m-1}]$ (where $[a_i, b_i]$ is the segment of the defensive line attacked by the i-th attacker) with the property that each sum $\sum_{j=a_i}^{b_i} v_j$ of defense values is equal to the attack strength k. The *value* of a strategy is the total number $\sum_{i=0}^{m-1}(b_i - a_i + 1)$ of attacked defenders. Find the maximum possible value of a legal strategy.

Input The first line of the input contains the number $t \leqslant 300$ of test cases. Each of the t test cases is described as follows.

- It starts with a line that contains three integers `n m k`, separated by a space. They denote
  - n, the number of players in the defensive line ($1 \leqslant n \leqslant 10^5$);
  - m, the number of attackers ($1 \leqslant m \leqslant 100$);
  - k, the attack strength ($1 \leqslant k < 10^9$).
- The following line defines the defense values. It contains n integers $v_0 \ \ldots \ v_{n-1}$, separated by a space, and such that $1 \leqslant v_i < 10^4$, for all $i \in \{0, \ldots, n-1\}$. Here v_i denotes the defense value of the i-th defender.

Output For each test case output one line that contains either a single integer that denotes the maximum value of a legal strategy, or the string `fail`, if no legal strategy exists.

Points There are six groups of test sets, worth 100 points in total.

1. For the first group of test sets, worth 20 points, you may assume $n \leqslant 300$ and $m = 2$.
2. For the second group of test sets, worth 15 points, and a corresponding hidden test set worth 5 points, you may assume $n \leqslant 3000$ and $m = 2$.
3. For the third group of test sets, worth 15 points, and a corresponding hidden test set worth 5 points, you may assume $m \leqslant 2$.
4. For the fourth group of test sets, worth 15 points, and a corresponding hidden test set worth 5 points, you may assume $n \leqslant 1000$.
5. For the fifth group of test sets, worth 15 points, and a corresponding hidden test set worth 5 points, there are no additional assumptions.

Corresponding sample test sets are contained in `testi.in/out`, for $i \in \{1, 2, 3, 4, 5\}$.

Sample Input

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

Sample Output

```
5
fail
5
```