第一题

1. 题目描述

Problem

There are **N** houses for sale. The i-th house costs **Ai** dollars to buy. You have a budget of **B** dollars to spend.

What is the maximum number of houses you can buy?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case begins with a single line containing the two integers **N** and **B**. The second line contains **N** integers. The i-th integer is **Ai**, the cost of the i-th house.

Output

For each test case, output one line containing `Case #x: y`, where `x` is the test case number (starting from 1) and `y` is the maximum number of houses you can buy.

Limits

Time limit: 15 seconds per test set. Memory limit: 1GB. 1 ≤ **T** ≤ 100. 1 ≤ **B** ≤ 105. 1 ≤ **Ai** ≤ 1000, for all i.

Test set 1

1 ≤ **N** ≤ 100.

Test set 2

1 ≤ **N** ≤ 105.

Sample

Sample

Input	Output
3 4 100 20 90 40 90 4 50 30 30 10 10 3 300 999 999 999	Case #1: 2 Case #2: 3 Case #3: 0

In Sample Case #1, you have a budget of 100 dollars. You can buy the 1st and 3rd houses for 20 + 40 = 60 dollars.

In Sample Case #2, you have a budget of 50 dollars. You can buy the 1st, 3rd and 4th houses for 30 + 10 + 10 = 50 dollars.

In Sample Case #3, you have a budget of 300 dollars. You cannot buy any houses (so the answer is 0).

2. 实现

简单送分题

```
#include <iostream>
#include <vector>
#include <set>
#include <queue>
#include <algorithm>
#include cmath
```

```cpp
using namespace std;
int main(){
    int t;
    cin >> t;
    for(int cases = 1; cases <= t; cases++){
        int n,b;
        cin >> n >> b;
        vector<int> A(n);
        for(int i = 0; i < n; i++)
            cin >> A[i];
        sort(A.being(), A.end());
        int cnt = 0, cost = 0;
        for(int i = 0; i < n; i++){
            cost += A[i];
            if(cost < b)
                ++cnt;
            else
                break;
        }
        cout << "Case #" << cases << ": " << cnt << endl;
    }
    return 0;
}
```

第二题

1. 题目描述

Problem

Dr. Patel has **N** stacks of plates. Each stack contains **K** plates. Each plate has a positive *beauty value*, describing how beautiful it looks.

Dr. Patel would like to take exactly **P** plates to use for dinner tonight. If he would like to take a plate in a stack, he must also take all of the plates above it in that stack as well.

Help Dr. Patel pick the **P** plates that would maximize the total sum of beauty values.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case begins with a line containing the three integers **N**, **K** and **P**. Then, **N** lines follow. The i-th line contains **K** integers, describing the beauty values of each stack of plates from top to bottom.

Output

For each test case, output one line containing `Case #x: y`, where `x` is the test case number (starting from 1) and `y` is the maximum total sum of beauty values that Dr. Patel could pick.

Limits

Time limit: 20 seconds per test set. Memory limit: 1GB. 1 ≤ **T** ≤ 100. 1 ≤ **K** ≤ 30. 1 ≤ **P** ≤ **N** * **K**. The beauty values are between 1 and 100, inclusive.

Test set 1

1 ≤ **N** ≤ 3.

Test set 2

1 ≤ **N** ≤ 50.

Sample

Sample

Input	Output
2 2 4 5 10 10 100 30 80 50 10 50 3 2 3 80 80 15 50 20 10	Case #1: 250 Case #2: 180

In Sample Case #1, Dr. Patel needs to pick **P** = 5 plates:

- He can pick the top 3 plates from the first stack (10 + 10 + 100 = 120).

- He can pick the top 2 plates from the second stack (80 + 50 = 130) .

In total, the sum of beauty values is 250.

In Sample Case #2, Dr. Patel needs to pick **P** = 3 plates:

- He can pick the top 2 plates from the first stack (80 + 80 = 160).
- He can pick no plates from the second stack.
- He can pick the top plate from the third stack (20).

In total, the sum of beauty values is 180.

2. 实现

动态规划的不完全背包问题啊啊啊啊啊，背包问题需要系统复习加强一下

```
#include<bits/stdc++.h>
using namespace std;
using LL = long long;
int dp[1500 + 1];//dp[i]表示在当前行之前的各行中取得i个盘子的最大beauty和
int b[30 + 1];
int main(){
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    int T;
    cin >> T;
    for(int t = 1; t <= T; t += 1){
        int N, K, P;
        cin >> N >> K >> P;
        fill(dp + 1, dp + P + 1, 0);
        for(int i = 1; i <= N; i += 1){
            for(int j = 1; j <= K; j += 1){
                cin >> b[j];
                b[j] += b[j - 1];
            }
            for(int j = P; j; j -= 1)
                for(int i = 1; i <= K and i <= j; i += 1)
                    dp[j] = max(dp[j], dp[j - i] + b[i]);//背包
        }
        cout << "Case #" << t << ": " << dp[P] << "\n";
    }
    return 0;
}
```

第三题

1. 题目描述

Problem

Tambourine has prepared a fitness program so that she can become more fit! The program is made of **N** sessions. During the i-th session, Tambourine will exercise for **Mi** minutes. The number of minutes she exercises in each session are *strictly increasing*.

The *difficulty* of her fitness program is equal to the maximum difference in the number of minutes between any two consecutive training sessions.

To make her program less difficult, Tambourine has decided to add up to **K** additional training sessions to her fitness program. She can add these sessions anywhere in her fitness program, and exercise any positive integer number of minutes in each of them. After the additional training session are added, the number of minutes she exercises in each session must still be strictly increasing. What is the minimum difficulty possible?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case begins with a line containing the two integers **N** and **K**. The second line contains **N** integers, the i-th of these is **Mi**, the number of minutes she will exercise in the i-th session.

Output

For each test case, output one line containing `Case #x: y`, where `x` is the test case number (starting from 1) and `y` is the minimum difficulty possible after up to **K** additional training sessions are added.

Limits

Time limit: 20 seconds per test set. Memory limit: 1GB. 1 ≤ **T** ≤ 100. For at most 10 test cases, 2 ≤ **N** ≤ 10^5. For all other test cases, 2 ≤ **N** ≤ 300. 1 ≤ **Mi** ≤ 10^9. **Mi** < **Mi+1** for all i.

Test set 1

K = 1.

Test set 2

1 ≤ **K** ≤ 10^5.

Samples

Samples

Input 1	Output 1
1 3 1 100 200 230	Case #1: 50

Input 2	Output 2
3 5 2 10 13 15 16 17 5 6	Case #1: 2 Case #2: 3

```
5 6                                    Case #2: 5
9 10 20 26 30                          Case #3: 1
8 3
1 2 3 4 5 6 7 10
```

Sample #1

In Case #1: Tambourine can add up to one session. The added sessions are marked in bold: 100 **150** 200 230. The difficulty is now 50.

Sample #2

In Case #1: Tambourine can add up to six sessions. The added sessions are marked in bold: 9 10 **12 14 16 18** 20 **23** 26 **29** 30. The difficulty is now 3.

In Case #2: Tambourine can add up to three sessions. The added sessions are marked in bold: 1 2 3 4 5 6 7 **8 9** 10. The difficulty is now 1. Note that Tambourine only added two sessions.

2. 实现

二分法，不知道怎么看出来的，大概是所有的nlong(n)算法里挨个试？或者从Mi的范围限制看出来的?

```
#include<bits/stdc++.h>
using namespace std;
using LL = long long;
constexpr int maxn = 120000;
LL M[maxn];
int main(){
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    int T;
    cin >> T;
    for(int t = 1; t <= T; t += 1){
        int N, K;
        cin >> N >> K;
        for(int i = 1; i <= N; i += 1) cin >> M[i];
        LL L = 1, R = 1000000000;
        while(L < R){
            LL m = (L + R) >> 1;//想要最大difficult为m，最少需要插入多少个训练
            LL k = 0;
            for(int i = 1; i < N; i += 1)
                k += (M[i + 1] - M[i] + m - 1) / m - 1;//需要在M[i]和M[i+1]之间插入的
训练
            if(k > K) L = m + 1;
            else R = m;
        }
        cout << "Case #" << t << ": " << L << "\n";
    }
```

```
    return 0;
}
```

第四题

1. 题目描述

Problem

Pip has **N** strings. Each string consists only of letters from `A` to `Z`. Pip would like to bundle their strings into *groups* of size **K**. Each string must belong to exactly one group.

The *score* of a group is equal to the length of the longest prefix shared by all the strings in that group. For example:

- The group `{RAINBOW, RANK, RANDOM, RANK}` has a score of 2 (the longest prefix is `'RA'`).
- The group `{FIRE, FIREBALL, FIREFIGHTER}` has a score of 4 (the longest prefix is `'FIRE'`).
- The group `{ALLOCATION, PLATE, WORKOUT, BUNDLING}` has a score of 0 (the longest prefix is `''`).

Please help Pip bundle their strings into groups of size **K**, such that the sum of scores of the groups is maximized.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case begins with a line containing the two integers **N** and **K**. Then, **N** lines follow, each containing one of Pip's strings.

Output

For each test case, output one line containing `Case #x: y`, where `x` is the test case number (starting from 1) and `y` is the maximum sum of scores possible.

Limits

Time limit: 20 seconds per test set. Memory limit: 1GB. 1 ≤ **T** ≤ 100. 2 ≤ **N** ≤ 105. 2 ≤ **K** ≤ **N**. **K** divides **N**. Each of Pip's strings contain at least one character. Each string consists only of letters from `A` to `Z`.

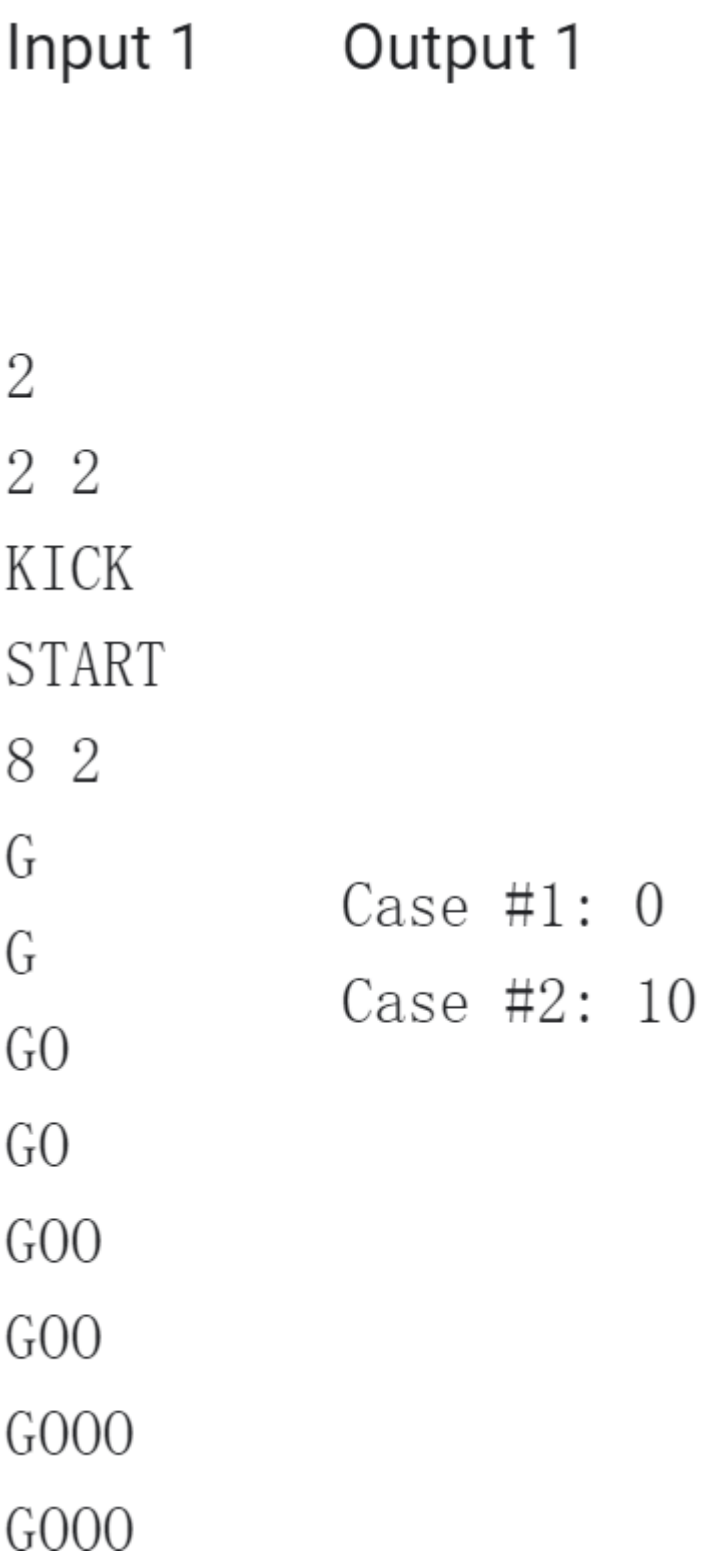
Test set 1

Each of Pip's strings contain at most 5 characters.

Test set 2

The total number of characters in Pip's strings across all test cases is at most 2 × 106.

Samples



Sample #1

In Case #1, Pip can achieve a total score of 0 by make the groups:

- `{KICK, START}`, with a score of 0.

In Case #2, Pip can achieve a total score of 10 by make the groups:

- `{G, G}`, with a score of 1.

- `{GO, GO}`, with a score of 2.
- `{GOO, GOO}`, with a score of 3.
- `{GOOO, GOOO}`, with a score of 4.

Input 2	Output 2
1 6 3 RAINBOW FIREBALL RANK RANDOM FIREWALL FIREFIGHTER	Case #1: 6

Sample #2

In Case #1, Pip can achieve a total score of 6 by make the groups:

- `{RAINBOW, RANK, RANDOM}`, with a score of 2.
- `{FIREBALL, FIREWALL, FIREFIGHTER}`, with a score of 4.

2. 实现

比赛时候看错题了卧槽，问的是每k个为一组，不是分成K组我FUCK，前缀树同时记录父节点、深度、单词数cnt（因为单词可以重复）即可

```
#include<bits/stdc++.h>
using namespace std;
using LL = long long;
constexpr int maxn = 2400000;
int kid[maxn][26], cnt[maxn], par[maxn], dep[maxn], nn;
int newnode(){//新建一个树节点
    nn += 1;//id
```

```
    fill(kid[nn], kid[nn] + 26, 0);
    cnt[nn] = par[nn] = dep[nn] = 0;
    return nn;
}
LL M[maxn];
int main(){
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    int T;
    cin >> T;
    for(int t = 1; t <= T; t += 1){
        nn = 0;
        newnode();
        int N, K;
        cin >> N >> K;
        for(int i = 0; i < N; i += 1){
            string s;
            cin >> s;
            int p = 1;
            for(char c : s){//将s加入前缀树
                if(not kid[p][c - 'A']){
                    kid[p][c - 'A'] = newnode();
                    par[kid[p][c - 'A']] = p;
                    dep[kid[p][c - 'A']] = dep[p] + 1;
                }
                p = kid[p][c - 'A'];
            }
            cnt[p] += 1;
        }
        int ans = 0;
        for(int i = nn; i; i -= 1){//一定是从某个叶子节点开始，逐渐向上遍历
            ans += cnt[i] / K * dep[i];//当前节点所能构成的组数*高度(公共前缀长度)
            cnt[par[i]] += cnt[i] % K;//剩余的不足K个的，将其加入父节点
        }
        cout << "Case #" << t << ": " << ans << "\n";
    }
    return 0;
}
```