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 m st 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 cont ining the three integers N, K and P. Then, N lines follow. The i-th line contains K integers, describing the beauty val es 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 th maximum total sum of beauty values that Dr. Patel could pick.

Limits

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

Test set 1

$1 \leq N \leq 3$.

Test set 2

$1 \leq N \leq 50$.

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.

Note: Unlike previous editions, in Kick Start 2020, all test sets are visible verdict test sets, meaning you receive inst nt feedback upon submission.

```
#include <bits/stdc++.h>
using namespace std;

#define ll long long
#define ar array

int n, k, p, a[50][30];
int dp[51][1501];

void solve() {
 cin >> n >> k >> p;
 memset(dp, 0xc0, sizeof(dp));
 dp[0][0]=0;
 for(int i=0; i<n; ++i) {
  memcpy(dp[i+1], dp[i], sizeof(dp[0]));
  for(int j=0, s=0; j<k; ++j) {
   cin >> a[i][j];
   s+=a[i][j];
   //use j+1 plates
   for(int l=0; l+j+1<=p; ++l)
    dp[i+1][l+j+1]=max(dp[i][l]+s, dp[i+1][l+j+1]);
  }
 }
 cout << dp[n][p] << "\n";
}

int main() {
 ios::sync_with_stdio(0);
 cin.tie(0);

 int t, i=1;
 cin >> t;
 while(t--) {
  cout << "Case #" << i << ": ";
  solve();
  ++i;
 }
}
```