Maximum subarray sum in an array created after repeated concatenation

> 给定数组arr，整数k，整数K代表将数组内的元素重复K次，即扩充后的数组arr的长度为 N * K，求子数组中的最大元素和

依然采用`kadane's algorithm`：

简单来讲就是，任意数组，比如说−2, 1, −3, 4, −1, 2, 1, −5, 4，其中的最大子列和，并不是容易的事情。但如果能从第一个数开始，随着数组的扩充，始终对其最大子列和保持跟踪，就可以轻易的求出任意一个数组的最大子列和。换言之，长度n的数组不容易求，长度为一是一定可以计算的，长度为一的算出来了，二也就能算出来，二算出来了，三就能算出来，以此类推，用这种根据i求i+1的思想，就能达到最终目的。

详细的分析一下，往一个长度为i的数组后面插入第i+1个数，`这时，数组的最大子列只有两种情况，要么包括第i+1个数，要么不包括第i+1个数`**。即：**
maxSubArrSum = max（以第i+1个数结尾的子列和， 不以第i+1个数结尾的子列和）

首先考虑`以第i+1个数结尾的子列和`：

要么它是以第i个数结尾的子列作为前缀，要么它不以之作为前缀。假设第i+1个数为x，即：以第i+1个数结尾的子列和 = max（x，以第i个数结尾的子列和 + x）

再考虑`不以第i+1个数结尾的子列和`：

其实就是插入第i+1个数之前的数组的最大子列和，而`插入第i+1个数之前的数组的最大子列和`的计算又回到了问题的最开始的情况，这其实是一个`数学归纳`。

算法实现：

定义两个变量：maxEndingHere代表到当前位置为止，所能够获得所有的子数组的最大的元素和，maxSoFarf用于记录最大的maxEndingHere

```
#include <bits/stdc++.h>

using namespace std;

int maxSubArraySum(vector<int> &arr, int &N, int &K) {
    int maxSoFar = arr[0], maxEndingHere = arr[0];
    for (int i = 1; i < N * K; ++i) {
        maxEndingHere = max(arr[i % N], maxEndingHere + arr[i % N]);
        maxSoFar = max(maxSoFar, maxEndingHere);
    }
    return maxSoFar;
}

int main() {
    int T;
    scanf("%d", &T);
    while (T--) {
        int N, K;
        scanf("%d %d", &N, &K);
        vector<int> arr(N);
        for (int i = 0; i < N; ++i)
            scanf("%d", &arr[i]);
        printf("%d\n", maxSubArraySum(arr, N, K));
    }
    return 0;

}
```