A

菲尼克斯有n个硬币，重2^1□2^2□...□2^n□他知道n是偶数。

他想把硬币分成两堆，这样每堆硬币正好有n/2个硬币，两堆硬币之间的重量差最小化。形式上□a表示第一堆中的权重之和□b表示第二堆中的权重之和。帮助菲尼克斯最小化a-b的绝对值。

输入

输入由多个测试用例组成。第一行包含整数t□1≤t≤100□-测试用例数。

每个测试用例的第一行包含一个整数n□2≤n≤30□n为偶数）-菲尼克斯拥有的硬币数量。

输出

对于每个测试用例，输出一个整数-两个桩之间可能的最小权重差。

样例

输入

2 2 4

输出

2 6

注意

在第一个测试案例中，凤凰有两个硬币，重量分别为2和4。无论他如何划分硬币，差别将是4-2=2。

在第二个测试案例中，凤凰有四枚重量分别为2、4、8和16的硬币。菲尼克斯最好把重量为2和16的硬币放在一堆里，把重量为4和8的硬币放在另一堆里。差别是（2+16）—（4+8）=6。

分析与解答

这个题很显然。如果将最大的2^n放到一堆，那么剩余所有的硬币加起来也不如2^n大。

因此，差别最小的，一定是最大的2^n和较小的-1+n/2个硬币放在一起，剩余硬币放在一起。根据等比数列求和，最小的差为2^{1+n/2}-2个。（赞同）

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

void solve(){
    int n;
    cin >> n;
    n/=2;
    long long int ans = pow(2, n + 1);
    cout <<ans - 2 << endl;
}

int main(){
```

```
  int t; cin>>t;
  while (t--)
    solve();
}
```

B

凤凰喜欢美丽的阵列。如果所有长度为k的子数列具有相同的和，则数组是美丽的。数组的子数列是任何连续元素的序列。

菲尼克斯现在有一个长度为n的数组a。他想在数组中插入一些整数，插入个数可能是0，这样数组就变得漂亮了。插入的整数必须介于1和n之间（包括1和n。。整数可以插入任何位置（甚至在第一个元素之前或之后），并且他并没有试图最小化插入的整数的数量。

输入

输入由多个测试用例组成。第一行包含整数t。1≤t≤50。-测试用例数。

每个测试用例的第一行包含两个整数n和k。1≤k≤n≤100。。

每个测试用例的第二行包含n个空格分隔的整数。1≤ai≤n。-Phoenix当前拥有的数组。这个数组可能很漂亮，也可能不漂亮。

输出

对于每个测试用例，如果不可能创建一个漂亮的数组，请打印-1。否则，打印两行。

第一行应包含美丽阵列的长度m。n≤m≤10^4。。你不需要最小化m。

第二行应该包含m个空格分隔的整数。1≤bi≤n。-菲尼克斯在其数组a中插入一些整数（可能为零）后可以获得的漂亮数组。您可以打印最初不在数组a中的整数。

如果有多种解决方案，请打印一个解决方案。它保证了如果我们能使数组变得漂亮，我们总是可以使它的长度不超过10^4。

样例

输入

4 4 2 1 2 2 1 4 3 1 2 2 1 3 2 1 2 3 4 4 4 3 4 2

输出

5 1 2 1 2 1 4 1 2 2 1 -1 7 4 3 2 1 4 3 2

注意

在第一个测试用例中，我们可以通过在索引3处插入整数1（在两个现有的2之间）使数组变漂亮。现在，长度k=2的所有子数组的和都是3。还有许多其他可能的解决方案，例如：

2,1,2,1,2,1 1,2,1,2,1,2

在第二个测试用例中，数组已经很漂亮了：长度为k=3的所有子数组都有相同的和5。 在第三个测试用例中，可以证明我们不能插入数字来使数组变得漂亮。 在第四个测试用例中，所示的阵列b是美丽的，并且长度k≥4的所有子阵列具有相同的和10。还有其他的解决办法。

分析

如果任意连续k个数都具有相同的和，那么这个数列一定是循环的，循环节为k。

这样一来，如果原数列中已经出现了超过k个不同的数字，一定不可能实现，否则应该可以实现。

既然不需要保证最短，我们只需要想各种办法让数列变成k位循环就行了。

假设这个数列一共用到了c个数字（c≤k）。首先从左往右读，直到c个数字均在数列中出现。假设这c个数字出现的次序依次是d1 d2……dc。

那么就可以构造一个循环数列，循环节为d1 d2……dc dc dc ……dc，共k位。

接下来再从头开始读。每当读下一个数字的时候，先判断是不是循环节中下一个数。如果不是，就把后面一整个循环节续进去，直到是为止。这样就构造完了这个冗长的数列。

（反正题目没有要求最优解，只要可行解就行了）

有c个数字（c≤k）的话，直接1，2，3，…k，行不？循环几遍到m>=n，停止？

也有道理。反正暴力解决，保证有子串就行了。

看了眼数据范围（n<100）这…直接输出自然数就行，要保证出现过，用个散列表统计一下。

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

#define maxn 1000000
#define max(a,b) a>b?a:b
int a[maxn];
int b[maxn];
int ans[maxn];

int t, n, i, j, k;
void solve(){
    cin >> n >> k;
    for(i = 0; i < n; i++){
        int tmp;
        cin >> tmp;
        a[tmp]++;
    }
    int cnt = 0;
    int max = 0;
    for(i = 1; i <= n; i++){
        if(a[i] > 0)    b[cnt++] = i;
    }
    if(cnt > k){
        cout << "-1" <<endl;
    }
```

```
    else{
        cout << n*k <<endl;
        while(n--){
            for(i = 0;i < k; i++){
                if (b[i]>0) cout<< b[i] << ' ';
                else cout << "1 ";
            }
        }
        cout << endl;
    }
    memset(a,0,maxn);
    memset(b,0,maxn);
    memset(ans,0,maxn);
}
int main(){
  int t; cin>>t;
  while (t--)
    solve();
}
```

C

菲尼克斯有一个由小写拉丁字母组成的字符串s。他想把字符串中的所有字母都分配到k个非空字符串a1，a2，…，ak中，这样s的每个字母都正好指向其中一个字符串ai。字符串ai不需要是s的子字符串。Phoenix可以分发s的字母，并在每个字符串ai中重新排列字母。

例如，如果s=baba和k=2，Phoenix可能会以多种方式分发字符串中的字母，例如：

ba和ba a和abb ab和ab aa和bb

但这些方法是无效的：

baa和ba b和ba baba和空字符串（ai应为非空）

菲尼克斯想把字符串s的字母分布成k个字符串a1，a2，…，ak，以最小化其中词汇最大字符串，即最小化max（a1，a2，…，ak）。帮助他找到最优分布并打印max（a1，a2，…，ak）的最小可能值。

String x的字形小于y，如果x是y的前缀并且x不等于y，或者存在下标i（1≤i≤min（|x|，|y|））使得xi<yi，对于每个j（1≤j<i）均有xj=yj。这里| x |表示字符串x的长度。

总之就是字典序的意思。

输入

输入由多个测试用例组成。第一行包含整数t（1≤t≤1000）-测试用例数。每个测试用例由两行组成。

每个测试用例的第一行包含两个整数n和k（1≤k≤n≤105）-字符串s的长度和非空字符串的数目。Phoenix希望将s的字母分别分配到这两个整数中。

每个测试用例的第二行包含一个长度为n的字符串s。该字符串仅由小写拉丁字母组成。

保证所有测试用例的n之和≤105。

输出

打印t答案-每个测试用例一个。第i个答案应该是第i个测试用例中max(a1,a2,...,ak)的最小可能值。

样例

输入

6 4 2 baba 5 2 baacb 5 3 baacb 5 3 aaaaa 6 4 aaxxzz 7 1 phoenix

输出

ab abbc b aa x ehinopx

注意

在第一个测试用例中，一个最优的解决方案是将baba分配到ab和ab。 在第二个测试用例中，一个最优的解决方案是将baacb分配到abbc和a中。 在第三个测试用例中，一个最佳解决方案是将baacb分配到ac,ab和b中。 在第四个测试用例中，一个最优的解决方案是将a aa aa分布到aa,aa和a中。 在第五个测试用例中，一个最佳解决方案是将aaxxzz分布到az,az,x和x中。 在第六个测试用例中，一个最佳的解决方案是将phoenix分配到ehinopx中。

分析

——以下算法好像出了点问题——

逆向考虑。

首先，把所有的字母全部拆掉，即对于长为n的串，先拆成n个单独的字母，然后不断拼接，直到拼成剩下k个串为止。

由于字母顺序可以打乱，那么先将n个字母按照字典序排序。（字母可重复）

每次合并的时候，把字典序最大的合并到字典序最小，并将新串串内排序，再插入原串的排列。这是二分查找加插入的操作。

大佬说用STL里面的multiset(元素可重复），本质还是树，每次取最大元和最小元维护就行了。

（瑟瑟发抖。至于具体怎么写，感觉有·点·复·杂）

——以上是有点问题的算法——

先排序，然后分这几种情况：

1. 比较s[0]和s[k-1]。如果不同直接输出s[k-1]。因为这时在首字母中s[k-1]已经是最大的了，后面什么都不加就是最小的，其他的加到别的地方。 2. 如果s[0]和s[k-1]相同，说明首字母相同，注意样例中的aaaaa。这就是一种特殊情况，这个情况a要均分，才是最小，比较s[k]和s[n-1]。相同就是尽量均分。 3. s[k]和s[n-1]不同的话，只能把s[k]到s[n-1]全都放在一个的后面，因为如果分给了其他的，一定会导致字典序增大。如abbc变成abc

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

```
void solve(){
    int n, k;
    cin >> n >> k;
    string s;
    cin >> s;
    sort(s.begin(),s.end());
    if(s[0]!=s[k-1]){
        cout<<s[k-1]<<endl;
        return;
    }
    cout<<s[k-1];
    if(s[k]!=s[n-1]){
        for(int i = k; i < n; i++){
            cout<<s[i];
        }
    }
    else{
        for(int i = 0; i < (n - 1) / k; i++){
            cout<<s[k];
        }
    }
    cout <<endl;
    return ;
}
int main(){
    int t;cin >> t;
    while(t--) solve();
}
```

D

菲尼克斯决定成为一名科学家！他目前正在调查细菌的生长。

最初，在第1天，有一种细菌的质量为1。

每天，一些细菌会分裂（可能为零或全部）。当一个质量为m的细菌分裂时，它变成两个质量为m/2的细菌。例如，一个质量为3的细菌可以分裂为两个质量为1.5的细菌。

而且，每天晚上，所有细菌的质量增加1。

菲尼克斯想知道所有细菌的总质量是否有可能精确到n。如果可能的话，他感兴趣的是如何使用尽可能少的夜数来获得这个质量。帮助他成为最好的科学家！

输入

输入由多个测试用例组成。第一行包含整数t（1≤t≤1000）-测试用例数。

每个测试用例的第一行包含一个整数n（2≤n≤109）-菲尼克斯感兴趣的细菌总数。

输出

对于每个测试用例，如果细菌无法精确达到总质量n请打印-1。否则，打印两行。

第一行应该包含一个整数d-所需的最少夜数。

下一行应该包含d个整数，第i个整数表示第i天应该分裂的细菌数量。

如果有多种解决方案，请打印任何解决方案。

例子

输入复制

3 9 11 2

输出

3 1 0 2 3 1 1 2 1 0

注意

在第一个测试用例中，以下过程产生总质量为9的细菌：

第1天：质量为1的细菌分裂。现在有两种细菌，每种质量为0.5。

夜1：所有细菌的质量增加一个。现在有两种质量为1.5的细菌。

第二天：没有分裂。

第二晚：现在有两种细菌的质量是2.5。

第三天：两种细菌都分裂了。现在有四种质量为1.25的细菌。

第三晚：现在有四种细菌的质量是2.25。

总质量为2.25+2.25+2.25+2.25=9。可以证明，3是所需的最少夜数。还有其他方法可以在3晚内获得总质量9。

在第二个测试用例中，以下过程产生总质量为11的细菌：

第1天：质量为1的细菌分裂。现在有两种质量为0.5的细菌。

夜1：现在有两种细菌的质量是1.5。

第二天：一个细菌分裂。现在有三种细菌的质量分别为0.75、0.75和1.5。

第二夜：现在有三种细菌，质量分别为1.75、1.75和2.5。

第3天：质量为1.75和2.5的细菌分裂。现在有5种细菌的质量分别为0.875、0.875、1.25、1.25和1.75。

第三夜：现在有五种细菌，它们的质量分别是1.875、1.875、2.25、2.25和2.75。

总质量为1.875+1.875+2.25+2.25+2.75=11。可以证明，3是所需的最少夜数。还有其他方法可以在3晚内获得总质量11。

在第三个测试案例中，细菌在第1天不会分裂，然后在第1天晚上生长到第2块。

分析

因为最后要求的是总质量，显然我们完全不用关心细菌究竟质量多少，只要看总个数就行了。相当于要求每次增加的质量为当前细菌个数，并且只增不减，要求从1开始，最少步骤精确达到给定值n

这个n就是细菌个数数列某个部分和+1（初始质量）。如果将1置于数列最前面（第0天），那就是部分和。要求数列下一项在前一项1倍到2倍之间。

可以想象，如果是2^n-1这样的数，只需要n-1次全体分裂就行了，也就是说次数的大致增长量级是对数量级。

那么方法也就呼之欲出了。

需要的最少晚上数，就是log_2 n即二进制位数-1。例如对于质量1，需要0个晚上。

细菌总个数和总质量无关……

对于二进制数1xxxxxxxx.....考虑后面xxxxxxxx.....部分：

从左往右数第一个1出现在哪里，代表现在正在修改第几位。

10000 0 1 2 4——初始数列0 1 2 4。

10001 1 0 2 4——从初始数列0 1 2 4到最终数列1 2 4 8过渡中，正在修改12两位。

10010 1 1 1 4——从初始数列0 1 2 4到最终数列1 2 4 8过渡中，正在修改23两位。 10011 1 2 0 4——从初始数列0 1 2 4到最终数列1 2 4 8过渡中，正在修改23两位。

10100 1 2 1 3——从初始数列0 1 2 4到最终数列1 2 4 8过渡中，正在修改34两位。 10101 1 2 2 2——从初始数列0 1 2 4到最终数列1 2 4 8过渡中，正在修改34两位。 10110 1 2 3 1——从初始数列0 1 2 4到最终数列1 2 4 8过渡中，正在修改34两位。 10111 1 2 4 0——从初始数列0 1 2 4到最终数列1 2 4 8过渡中，正在修改34两位。

11000 1 2 4 1——修改1 2 4 8的末位。 …… 11111 1 2 4 8

100000 0 1 2 4 8——以此类推

被修改的两位，和始终保持不变，为2的{位数减一}次幂，前一个数是标红的1后面部分减1。

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

using namespace std;
void solve(){
    vector<int>inc;
    int N;
    cin >> N;
    for(int i = 1; i <= N; i*=2){
        inc.push_back(i);
        N-=i;
    }
    if(N>0)    inc.push_back(N);
    sort(inc.begin(),inc.end());
    cout << inc.size()-1<<endl;
    for(int i = 1; i<inc.size(); i++){
        cout<<inc[i]-inc[i-1]<<' ';
    }
    cout <<endl;
```

```
}
int main(){
    int t; cin >> t;
    while(t--)
        solve();
}
```

E

菲尼克斯在后院摘浆果。有n灌木，每个灌木有a_i红色浆果和b_i蓝色浆果。

每个篮子可以装k浆果。但是，菲尼克斯决定每个篮子只能装同一种灌木的浆果或同一颜色的浆果（红色或蓝色）。换句话说，篮子里的所有浆果必须来自同一种灌木或/和具有相同的颜色。

例如，如果有两个灌木，第一个灌木中有5红色和2蓝色浆果，第二个灌木中有2红色和1蓝色浆果，则Phoenix可以完全填充2篮容量4：

第一个篮子将包含来自第一个灌木的3红色和1蓝色浆果；

第二个篮子将包含来自第一个灌木的2剩余红色浆果和来自第二个灌木的2红色浆果。

帮助菲尼克斯确定他能完全填满的最大篮数！

输入

第一行包含两个整数n和k□1\le n□k\le 500□-分别是灌木的数量和篮子容量。

接下来的n行中的i-th包含两个整数a_i和b_i□$$$0\le a_i□b_i\le 10^9$$□-分别是i-th灌木中红色和蓝色浆果的数量。 输出 输出一个整数-菲尼克斯可以完全填充的最大篮数。 样例 输入 2 4 5 2 2 1 输出 2 输入 1 5 2 3 输出 1 输入 2 5 2 1 1 3 输出 0 输入 1 2 1000000000 1 输出 500000000 注意 上面描述了第一个示例。 在第二个例子中，菲尼克斯可以用第一个（也是唯一一个）灌木的所有浆果填满一个篮子。 在第三个示例中□Phoenix无法完全填充任何篮子，因为每个灌木中的浆果少于$$$5$$，总红色浆果少于$$$5$$，总蓝色浆果少于$$$5$$。

在第四个例子中，菲尼克斯可以把所有的红色浆果放进篮子里，留下一个额外的蓝色浆果。

分析

（暂无思路）

大致意思是只能从矩阵同行或同列取值，要求每次取值必须填满篮子□k个），问最多填多少个篮子。

F

菲尼克斯正在给他的n个朋友拍照，他们的标签是1□2□...□n□按特殊顺序排成一排。但他还没来得及拍照，他的朋友们就被一只鸭子弄得心烦意乱，把秩序搞得一团糟。

现在，菲尼克斯必须恢复秩序，但他记不清了！他只记得左边的第i个朋友在ai和bi-inclusive之间有一个标签。有没有一种独特的方法可以根据他的记忆来安排他的朋友？

输入

第一行包含一个整数n（1≤n≤2·105）-朋友数。

接下来n行中的第i行包含两个整数ai和bi（1≤ai≤bi≤n）-菲尼克斯从左侧对第i个位置的记忆。

菲尼克斯的内存是有效的，所以至少有一个有效的顺序。

输出

如果菲尼克斯可以按照唯一的顺序重新排列他的朋友，请在“ 是 ”后面加上n个整数-第i个整数应该是第i个朋友在左侧的标签。

否则，请打印“ 否 ”。然后，在以下两行上打印任意两个不同的有效订单。如果是多个解决方案，请打印任意两个。

样例

输入

4 4 4 1 3 2 4 3 4

输出

YES 4 1 2 3

输入

4 1 3 2 4 3 4 2 3

输出

NO 1 3 4 2 1 2 4 3

分析

（这个题也没思路）

大致意思是找一个合法的排列，使得每个数恰好落进给定的区间。

如果合法排列唯一，是YES（合法排列数大于等于2）NO并输出任意两个。

From:
https://wiki.cvbbacm.com/ - **CVBB ACM Team**

Permanent link:
https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:namespace:codeforces_round_638_div._2&rev=1590928562

Last update: **2020/05/31 20:36**