Fruit Ninja

明显与 5 的倍数有关，于是分类讨论：
一、如果 m 不是 5 的倍数，显然加分可以很快结束，直接模拟就行了
二、如果 m 是 5 的倍数的话
I、如果 n 是 5 的倍数，那么答案肯定是 INF，只要原始得分是 5 的倍数
II、n 不是 5 的倍数
1、对于当前分数，如果是 5 的倍数，那么肯定是分数一直+m 直到
到达 55，555 这类的数，最后在加 m+n，然后即两个条件都不满足，加分结束。(看到很多 wa 的都直接判断 m%5==0 就是 INF，应该是没考虑到这种情况)
因为题目保证答案不超 long long，那么只需对 5，55，555，……5555555555555555555 逐一验证，如果这几个数都达不到的话那么，最后肯定是 INF 。
2、如果当前分数不是 5 的倍数，直接模拟，直到分数为 5 的倍数，转到 1

要做出这道题，只要知道上面几点就行了

下面继续说明，为什么达不到 5，55，555，……5555555555555555555 这几个数就是 INF

设 m=5*k ， 分数为 5*p
那么达到那几个数的条件是 5*k*t+5*p=555……55 ,t 代表第一个条件加分加的次数
化简就是 k*t+p=111……11
这个方程满足的条件是 111……111 和 p 关于 k 同余
如果答案不是 Inf 的话，只有形如 111……111 这类的数模上 k 能取遍 0~k-1（因为 p 模 k 能取到 0~k-1）
因为 m<=50 ,所以 k 只能取 1，2，3，4，5，6，7，8，9，10
那么模 k 最多有 10 个余数,而在 longlong 内形如 111……111 这类的数有 19 个，所以如果这 19 个数
模上 k 不能取遍 0~k-1 的话，那么肯定已经出现了循环，数再大也不行了，肯定是 INF

Game

先求出 Bob 最后值为各个数的概率。之后再对于 Bob 最后的每个值 x，求出 Alice 最后值大于 x 的概率和等于 x 的概率。
其实只有一个问题，即对于一个给定的 x，求出区间长度大于等于 len 且 区间第 len 小的值大于 x 的个数。
因为等于 x 的情况就是大于 x-1 的区间个数 减去 大于 x 的区间个数。而概率就是区间个数除以总的合法区间个数。

对于这个问题，有个类似滑动窗口的做法。首先数列的每个数只有两种性质，大于 x，或者小于等于 x。

枚举选取区间的左端点 i，可以得到从某个 j（包括 j)之后的所有右端点区间第 len 小值都不超过 x，而满足的这个 j 就是
j=min{j|sigma((bool)a[k]<=x)>=len,i<=k<=j};
则此时满足要求区间个数为 sigma(max(0,j-i-len+1));由于 j 随着 i 的增加而不减，因此可以在 O(n)的时间内完成。
本题总的复杂度是 O(n^2).

Hongshu

这道题由于数据范围保证了在 10000 年之内一定能够完成所有工作，因此我们可以通过枚举时间的方式对每项任务求出对应的时间，算法的大致流程如下：
1、令当前时间为 2001 年 1 月 1 日 10 点
2、循环读入每个任务请求
a、如果当前时间小于任务送达的时间，就把当前时间设为满足下面几个条件的最早的时刻，否则当前时间不变。
-首先必须在星期一到星期五，其次必须在 10 点到 18 点之间，而且必须大于等于任务送达的时间
b、按照题目要求，求出 Hongshu 完成任务时的时间（也就是工作 T 小时之后的时刻，同样需要注意不要在双休日工作~）
c、输出答案，并将当前时间定为完成任务的时间

枚举有两种方法
1、按小时枚举，比较暴力，但考虑的特殊情况相对较少，考虑了星期五的情况之后，还需要注意一个地方：
如果当前是某天的 18 点，那么再工作一个小时之后是次日的 11 点
如果是为了求“大于等于任务送达时间的最早的时刻”，那么从某天的 18 点出发，下一个需要枚举的是次日的 10 点。
2、按天枚举，其实就是每天工作 8 个小时，一共 T 个小时，求一共需要工作几天之类的问题，其实和小时枚举差不多，嗯，大约就是 20msAC 和 200msAC 的区别，无关大局。

另外输入是按照时间顺序来的，如果读入又排了个序的话，对于同一时刻的多个任务处理不当的话可能会挂的哟。

Improving Roads

解法 1：我们定义用 Dijkstra 算法扩展出的生成树为最短路生成树。本题就是求最小的最短路生成树。回忆 Dijkstra 算法的过程，维护已确定最短路的点集 S 和未确定最短路的点集 T，每次从 T 中找出距离值最小的点 u 加入 S，并用 u 去更新 T 中的点的距离值。多解出现在，把 u 加入生成树的边可能有多个。然而加最小的边肯定是最优的。因为一旦加入生成树，对后面没有任何影响，所以当前最优即可保证全局最优。
解法 2：由条件 1 可知，最后答案的每条边肯定在从起点到某个点的某条最短路上。因而可以先求出满足这个要求的所有边。即先用 dijkstra 或者 spfa 求

出起点到每个点的最短路 dis[i],接着对于原图中的任何一条边(u,v,w),假如 dis[u]+w=dis[v],则这条边在从起点到点 v 的某条最短路上。所有满足的这些边构成一个新的图 G'=(V',E').

根据条件 2，就是在新图 G'上求最小树形图(因为是有向边)。但仔细分析可以注意到新图 G'是一个拓扑图，用 ans[v]=min{w|(u,v)∈E'}，则最小生成树的边权总和 sum=Σans[i](1<i<=n).理由如下：除了起点，每个点都是取的它最小的入边，由于是最小生成树，入边是必须的，因而最优值不可能小于这个值。由于这些点都有入边，因而他们必然能如某个入度为 0 的点到达，而入度为 0 的点就只有起点，因而所有这些点是相连的。

Jack and Rose

首先 Jack 取的是左边的一段，Rose 取的是右边的一段，所以可以把在每个点相遇，两个人的胜负情况预处理出来。然后分类讨论：

a.偶数个 slot：记中间两个 slot 为 k, k+1；

1、要是在 k 相遇 Jack 能够赢，那么第一步 Jack 将 B 往左移动一步，以后 Jack 的选择是上一步 Rose 移动棋子的另外一个，即和 Rose 相反，那么总能够在 k 相遇；

2、要是在 k+1 相遇 Jack 能够赢，那么第一步 Jack 将 A 往右移动一步，以后 Jack 的选择是上一步 Rose 移动棋子的另外一个，即和 Rose 相反，那么总能够在 k+1 相遇；

3、如果两个点都是 Jack 输的话，Rose 每一步都和 Jack 选择相反的棋子移动，最终将在 k 或者 k+1 相遇，Jack 将输给 Rose；

4、最后如果两个点有一个是 Jack 输，一个是平局（或两个都是平局）按照上面的策略 Jack 可以保证在平局的那个点相遇，当然 Rose 就更不可能输了，所以结果是平局。

b.奇数个 slot：记中间那个 slot 为 k；

1、在 k 点 Jack 输，Rose 每一步都和 Jack 选择相反的棋子移动，那么将保证最终他们在 k 相遇，所以，Rose 就赢了；

2、在 k 点是平局，那么只要 Rose 还是按照情况 1（在 k 点 Jack 输）的策略 Rose 至少不会输，那么只要看 Rose 怎么赢

1）在 k-1 和 k+1 相遇都是 Jack 输，那么 Rose 在最后一步之前，都和 Jack 选择相反的棋子，最后一步和 Jack 选择相同的棋子，Rose 也将获胜（要么在 k-1 相遇了，要么在 k+1 相遇了）；

2）否则（在 k-1 和 k+1 Jack 不都是输），那么现在的情况是 k-1 与 k ，k 与 k+1 ，这两组至少有一组 Jack 在两个点都不输，我们不妨设是 k 与 k+1 ， Jack 第一步可以把 A 往右移（如果是 k-1 与 k，把 B 往左移），那么剩下的 slot 就是偶数个，并且中间点是 k 和 k+1，参照上面偶数第三种情况，以后 Jack 每步都选择和 Rose 相反的棋子移动，最终在 k 或 k+1 相遇，所以 Jack 不会输，这样结果就是平局了。

3、在 k 点是 Jack 赢

1）在 k-1 和 k+1 相遇都是 Jack 输，Rose 采取情况 2（在 k 点平局）的第一种情况的策略，Rose 赢。

2）在 k-1 和 k+1 点有一个是 Jack 赢，只要 Jack 采取情况 2 的第二种情况的策略，Jack 就赢了。

3）在 k-1 和 k+1 一个是 Jack 输，一个点是平局(或都是平局)，Rose 采取情况 2 第一种情况的策略，Jack 采取情况 2 的第二种情况的策略，两者都不会输，结果是平局。

PS:
大家可以发现偶数只和中间两个点有关，奇数只和中间三个点有关，所以怎么写简单点，你懂的。
看了 AC 的程序，都是判断如果两者必胜策略都不存在就是 tie，但这样还是有点不严谨，虽然事实确实如此，因为根据上面的分析只要自己必胜策略不存在，对方就一定存在不输的策略，标程也是这样写的。

KKX Sequence

对于序列的一种排列 A[1], A[2], ... A[N]；

最后的结果为 C[N-1][0] * A[1] - C[N-1][1] * A[2] + C[N-1][2] * A[3] ... + ((-1)^k) * C[N-1][k] * A[k+1] + ... + ((-1)^(N-1)) * C[N-1][N-1] * A[N]；

其中 C[N-1][i]代表组合数 C(n-1,i)，这个可以通过递推得到（Hint：杨辉三角）

上述公式可以用数学归纳法证明；

所以，我们将 ((-1)^k) * C[N-1][k], k = 0, 1, ... N-1. 按照从小到大排序；

再将 A[k], k = 0, 1, ... N. 按照从小到大排序；

最后将两个序列求内积（对应位相乘后相加）就是最终结果。

另外，最后的结果最大为 (2 ^ 49) * 1000，在 long long 范围内。

关于为什么排序之后得到的结果一定是最优的，同样可以通过数学归纳法予以证明，对于长度为 2 的数组，结论显然成立，而关于归纳法的第二步，如果对于长度为 2~k 的数组这个结论都成立，对于长度为(k+1)的数组结论为什么一定成立，希望大家仔细思考一下。