A：
简单极角排序。
对每条线段的端点进行极角排序。对于检查到进入端点，统计数目加一；对于检查到结束端点，统计数目减一。
对于跨过-x 轴方向的线段，可以在±π的位置添加端点，从而减少特判的情况。

B：
模拟游戏扫雷的标号规则，不过要注意输出要求。

C：
DP，01 背包问题。
先计算出最多可以花费在点心上面的金额是多少，然后将每种点心的 favour 值加起来作为点心价值。每种点心最多只能有两个，因为只有两个，所以可以不用多重背包。最后求出最优值后求平均值即可。

D：
积分+二分。
因为温度下降速率与当前温度值有关，所以可以得到一个函数 f(t)，对其积分。
对于求某个时刻的温度可以直接代入公式；相反，求时间的时候可以通过二分判断。

参数的计算：`k = t1 / log((u0 - ua) / (u1 - ua));`

```
double cal(double x) { return k * log((u0 - ua) / (x - ua));}
```

E：
KMP 找最短循环节+Polya 定理（入门级）。
对于 Polya 定理，与 POJ 1268 的计算方法差不多。

F：
差分+BIT 维护。

首先从一维的差分进行理解：
一维 BIT（区间求和，区间更新）
d[i]=a[i]-a[i-1]（差分数组）
sigma{a[i]}=(n+1)*sigma{d[i]}-sigma{d[i]*i}

```
struct BIT2 {
    BIT a, b;
    void init(int n) { a.init(n), b.init(n);}
    void add(int x, LL d) { a.add(x, d), b.add(x, x * d);}
    void add(int x, int y, LL d) { add(x, d), add(y + 1, -d);}
    LL sum(int x) { return (x + 1) * a.sum(x) - b.sum(x);}
    LL sum(int x, int y) { return sum(y) - sum(x - 1);}
} bit;
```

对于扩展到二维的情况，我们可以，令
a[i][j]——(i,j)-(n,m)增量（差分子矩阵）
S[i][j]——(1,1)-(i,j)求和（这部分在理解一维情况后，自主推导）

G：
Hash+BFS。
这题需要比较强的代码组织能力，不然较长的代码容易造成很多注意不到的错误。
这题可以通过地图只有 10*10，然而只有 8*8 是有用的，于是可以判断出来可对两个 Robot 的位置及面向的方向进行 Hash，同时把 Pincer 的位置也 Hash 进去。
每个 Robot 状态只有 8*8*4，Pincer 的状态只有 8*8，于是总的状态数实际只有(1 << 22)种。
做这题的时候，因为留意到 Robot 是根据人的位置自动的转向的，所以可以给机器人写一个自动移动的类，同时写一个 Hash 值的输入输出接口即可。
Bfs 的时候需要同时记录前驱，以便最后回溯答案。