Solution　8.1

T1　夹角

签到题

因为 x 可能会大于12，所以我们可以先预处理 $x \mod = 12$，先求时针和分针之间的顺时针角度：$ans = fabs((x + \frac{y}{60}) \cdot 30 - 6 \cdot y)$;

可能之间的逆时针距离角度，所以 $ans = min(ans, 360 - ans)$;

核心代码：

```
ans=fabs((double)(x+y/60.0)*30-6.0*(double)y);//求顺时针角度
```

```
ans=min(ans,360.0-ans);//取顺时针角度和逆时针角度的最小值
```

T2 加密

核心算法：二分

q表示在 $[1, r]$ 的区间里的第 q 个数(并非编号)，因为每次都会相反，所以每次二分开始时为 $l + r - q$；

如果 $q \leq mid$ 即为 $r = mid$，否则为 $l = mid + 1$

解释一下为什么是 $l + r - q$。

很显然是 $[l, r]$ 中的第 $r - q + 1$ 个数，然后还要加上 $l - 1$，即为 $l + r - q$

核心代码

```
while(l<r)//二分
  {
    q=l+r-q;// [1,r]的区间里的第q个数(并非编号)
    mid=l+r>>1;
    if(q<=mid)
      r=mid;
    else
      l=mid+1;
  }
```

T3

难度评价：普及+/提高

算法一

对于$k=1$且a_i全为正数的数据，可以发现这就是一个经典的01背包问题。

时间复杂度$O(nm)$，空间复杂度$O(m)$，期望得分25。

算法二

对于a_i全为正数的数据，可以发现这就是一个经典的多重背包问题。暴力或二进制拆分有不同的部分分。

暴力时间复杂度$O(nmk)$，空间复杂度$O(m)$，期望得分40。

二进制拆分时间复杂度$O(nm\log k)$，空间复杂度$O(m)$，期望得分55。

算法三

对于a_i可能为负数的数据，01背包和多重背包之前的调整DP顺序对空间的优化需要进行一些改进。

经典的01、多重背包中，a_i是正数，为保证第i天不会重复转移多次，DP的转移顺序为倒序。

但如果a_i是负数，为保证第i天不会重复转移，DP的转移顺序应为正序。

此外，数组下标不能是负数，所以可将$[-m,m]$统一向右平移m。

时间复杂度$O(nm\log k)$，空间复杂度$O(m)$，期望得分100分。

如果对转移顺序不清楚，可以直接多加一维表示当前天数，空间复杂度$O(nm)$，期望得分75分。用滚动数组优化也可得到100分。

T4 求和

我们枚举 i 作为约数出现的次数，所以原问题等价于 $\sum_{i=1}^{n}\frac{n}{i}$

给定 $1 \leq n, k \leq 10^9$ 求

$$\sum_{x=1}^{n} \lfloor \frac{k}{x} \rfloor$$

其中 $\lfloor x \rfloor$ 是指实数 x 的整数部分。

我们可以直接根据题意，写出一个循环语句，代码如下

```
for (int i = 1; i <= n; ++i) ans += k / i;
```

容易得知，这样做的时间复杂度是 $O(n)$ 的。而 n 的范围过大，所以需要对这个算法进行优化。

【引理1】对于 $g(x) = \lfloor \frac{n}{x} \rfloor$（其中 x 为正整数，且 $1 \leq x \leq n$ ），则 g(x) 不同值的个数不会超过 $2\sqrt{n}$ 个。

证明:

可以将 $g(x)$ 的值分成小于 $\sqrt{n}$ 和大于等于 $\sqrt{n}$ 的两部分，如下：

（1）当 $g(x) < \sqrt{n}$ 时，$g(x)$ 最多 $\sqrt{n} - 1$ 个值。

（2）当 $g(x) \geq \sqrt{n}$ 时，则 $\sqrt{n} \leq \lfloor \frac{n}{x} \rfloor \leq \frac{n}{x}$，从而推导得出 $x \leq \sqrt{n}$，所以这种情况下 x 范围为 $[1, \sqrt{n}]$，即最多 $\sqrt{n}$ 个值，自然 $g(x)$ 也就最多 $\sqrt{n}$ 个值。

综上所述，g(x) 的值不会超过 $2\sqrt{n}$ 个值。

x	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$\lfloor \frac{n}{x} \rfloor$	25	12	8	6	5	4	3	3	2	2	2	2	1	1	1	1

核心代码：

```
for (int l = 1, r; l <= n; l = r + 1) {
    r = n / (n / l); ans += (r - l + 1) * (n / l);
}
```