



# A ICPC Penalty



判断后直接计算即可，注意 *Accepted* 时  $n$  次提交有  $n - 1$  次罚时。

```
1 void solve() {
2     string s;
3     int n, t;
4     cin >> s >> n >> t;
5     if (s == "Accepted") {
6         cout << 20 * (n - 1) + t << "\n";
7     } else {
8         cout << 0 << "\n";
9     }
10 }
```

Fence 1



## B 小笨的无限数组



第一次筛去所有奇数；

第二次相当于对第一次的结果除以 2 后筛去所有奇数，剩下 4 的倍数。

以此类推，第  $n$  次剩下的数字就是  $2^n$  的倍数。

```
1 vector<int> p2(11);
2
3 void init() {
4     p2[0] = 1;
5     for (int i = 1; i <= 10; ++i) p2[i] = p2[i - 1] + p2[i - 1];
6 }
7
8 void solve() {
9     int n, k;
10    cin >> n >> k;
11    cout << p2[k]*n << "\n";
12 }
```

Fence 2

当然也可以直接打表找规律

```
1 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
2 2 4 6 8 10 12 14 16 18 20
3 4 8 12 16 20
4 8 16
5 16
```

Fence 3



## C 小笨的真假游戏



记  $x_i$  为第  $i$  位是否说真话， $s_i$  为第  $i$  对第  $(i - 1 + n)$  位是否说真话的断言。

- $x_i = 1$  ,  $s_i = x_{i-1}$
- $x_i = 0$  ,  $s_i = 1 - x_{i-1}$

也就是  $s_i = x_{i-1} \oplus (1 - x_i)$

记  $t_i = s_i \oplus 1$  (取反) , 如果一个串是合法的, 必然有

$$x_1 = x_1 \oplus (t_1 \oplus t_2 \oplus \cdots \oplus t_n)$$

$$t_1 \oplus t_2 \oplus \cdots \oplus t_n = 0$$

即只有偶数数个  $i$  满足  $t_i = 1$  , 也就是字符串只有偶数个 0 。

然后对  $x_1$  赋值 0 和 1 即可得到唯二的两种合法情况。

```
1 void solve() {
2     int n, cnt = 0;
3     string s;
4     cin >> n >> s;
5     for (auto v : s) cnt += v == '0';
6     cout << (cnt & 1 ? 0 : 2) << "\n";
7 }
```

Fence 4



# D 小笨的GCD疑问1.0

猜一个全选就是最优解，举个例子：

$$3 + 3 + 3 + 6 + 6 + 6 \leq 3 + 4 + 5 + 6 + 7 + 8$$

```
1 void solve() {
2     ll l, r, k;
3     cin >> l >> r >> k;
4     ll n = r - l + 1;
5     if (n == 1) {
6         cout << 0 << "\n";
7         return;
8     }
9     cout << (l + r) * n / 2 - r << "\n";
10 }
```

Fence 5

当然也可以暴力，因为涉及一些编译优化，所以将完整代码放到题解最后。



## E 小苯的区间选数2.0



收集所有区间然后贪心放置即可，先用左小长度小的区间。

```
1 void solve() {
2     int n;
3     cin >> n;
4     vector<PLL> a(n);
5     for (int i = 0; i < n; ++i) cin >> a[i].fi >> a[i].se;
6     sort(all(a));
7     priority_queue<ll> pq;
8     int i = 0;
9     ll cnt = 0;
10    while (1) {
11        while (i < n && a[i].fi <= cnt) {
12            pq.push(-a[i].se);
13            ++i;
14        }
15        while (!pq.empty() && pq.top() < cnt) pq.pop();
16        if (pq.empty()) break;
17        pq.pop();
18        ++cnt;
19        if (cnt > n) break;
20    }
21    cout << cnt << "\n";
22 }
```

Fence 6



## F 小苯的GCD疑问2.0

枚举  $gcd$  即可，感觉比  $D$  简单。

```
1 void solve() {
2     ll n, k;
3     cin >> n >> k;
4     ll g = 1, ans = 0;
5     while (g <= n / k) {
6         ll m = n / g;
7         ll ng = min(n / m, n / k);
8         ans = max(ans, ng * ng * m * (m + 1) / 2);
9         g = ng + 1;
10    }
11    cout << ans << "\n";
12 }
```

Fence 7



# 头文件

```
1 //Another
2 #include<bits/stdc++.h>
3 #include<bits/extc++.h>
4 #define pb push_back
5 #define eb emplace_back
6 #define fi first
7 #define se second
8 #define all(a) a.begin(), a.end()
9 #define rall(a) a.rbegin(), a.rend()
10 using namespace std;
11
12 typedef long long ll;
13 typedef long double ld;
14 typedef unsigned long long ull;
15 typedef __int128 i128;
16 typedef pair<int, int> PII;
17 typedef pair<ll, ll> PLL;
18 typedef tuple<ll, ll, ll> TLLL;
19 typedef __gnu_pbds::tree<PLL, __gnu_pbds::null_type,
20 less<PLL>, __gnu_pbds::rb_tree_tag,
21 __gnu_pbds::tree_order_statistics_node_update> Tree;
22 // typedef __gnu_pbds::tree<ll, __gnu_pbds::null_type,
23 less<ll>, __gnu_pbds::rb_tree_tag,
24 __gnu_pbds::tree_order_statistics_node_update> Tree;
25
26 constexpr int inf = (ll)1e9 + 7;
27 constexpr ll INF = (ll)2e18 + 9;
28 // constexpr ll INF = (ll)4e18;
29 // constexpr ll MOD = 1e9 + 7;
30 constexpr ll MOD = 998244353;
31 constexpr ld PI = acos(-1.0);
32 constexpr ld eps = 1e-10;
33
34 mt19937_64
35 rng(chrono::steady_clock::now().time_since_epoch().count());
36 ull randint(ull l, ull r) {uniform_int_distribution<unsigned
37 long long> dist(l, r); return dist(rng);}
```

```
32
33 void init() {
34
35 }
36
37 void solve() {
38
39 }
40
41 int main() {
42     ios::sync_with_stdio(0);
43     cin.tie(0); cout.tie(0);
44
45     init();
46
47     int t = 1;
48     // cin >> t;
49     for (int _ = 1; _ <= t; ++_) {
50         solve();
51     }
52     return 0;
53 }
```

Fence 8



## D题暴力



● ● ●

```
1 #include <bits/stdc++.h>
2 #pragma GCC optimize(3)
3 #pragma GCC target("avx")
4 #pragma GCC optimize("Ofast")
5 #pragma GCC optimize("inline")
6 #pragma GCC optimize("-fgcse")
7 #pragma GCC optimize("-fgcse-lm")
8 #pragma GCC optimize("-fipa-sra")
9 #pragma GCC optimize("-ftree-pre")
10 #pragma GCC optimize("-ftree-vrp")
11 #pragma GCC optimize("-fpeephole2")
12 #pragma GCC optimize("-ffast-math")
13 #pragma GCC optimize("-fsched-spec")
14 #pragma GCC optimize("unroll-loops")
15 #pragma GCC optimize("-falign-jumps")
16 #pragma GCC optimize("-falign-loops")
17 #pragma GCC optimize("-falign-labels")
18 #pragma GCC optimize("-fdevirtualize")
19 #pragma GCC optimize("-fcaller-saves")
20 #pragma GCC optimize("-fcrossjumping")
21 #pragma GCC optimize("-fthread-jumps")
22 #pragma GCC optimize("-funroll-loops")
23 #pragma GCC optimize("-fwhole-program")
24 #pragma GCC optimize("-freorder-blocks")
25 #pragma GCC optimize("-fschedule-insns")
26 #pragma GCC optimize("inline-functions")
27 #pragma GCC optimize("-ftree-tail-merge")
28 #pragma GCC optimize("-fschedule-insns2")
29 #pragma GCC optimize("-fstrict-aliasing")
30 #pragma GCC optimize("-fstrict-overflow")
31 #pragma GCC optimize("-falign-functions")
32 #pragma GCC optimize("-fcse-skip-blocks")
33 #pragma GCC optimize("-fcse-follow-jumps")
34 #pragma GCC optimize("-fsched-interblock")
35 #pragma GCC optimize("-fpartial-inlining")
36 #pragma GCC optimize("no-stack-protector")
37 #pragma GCC optimize("-freorder-functions")
```

```
38 #pragma GCC optimize("-findirect-inlining")
39 #pragma GCC optimize("-fhoist-adjacent-loads")
40 #pragma GCC optimize("-frerun-cse-after-loop")
41 #pragma GCC optimize("inline-small-functions")
42 #pragma GCC optimize("finline-small-functions")
43 #pragma GCC optimize("-ftree-switch-conversion")
44 #pragma GCC optimize("-foptimize-sibling-calls")
45 #pragma GCC optimize("-fexpensive-optimizations")
46 #pragma GCC optimize("-funsafe-loop-optimizations")
47 #pragma GCC optimize("inline-functions-called-once")
48 #pragma GCC optimize("-fdelete-null-pointer-checks")
49 #pragma GCC optimize(2)
50 using namespace std;
51 using ll = long long;
52
53 static const size_t IN_BUF_SZ = 1 << 20;
54 static char inbuf[IN_BUF_SZ];
55 static size_t in_pos = 0, in_len = 0;
56
57 static inline int getchar_()
58 __attribute__((always_inline));
59 static inline int getchar_() {
60     if (in_pos >= in_len) {
61         in_len = fread(inbuf, 1, IN_BUF_SZ, stdin);
62         in_pos = 0;
63         if (in_len == 0) return EOF;
64     }
65     return inbuf[in_pos++];
66 }
67
68 static inline ll rd() __attribute__((always_inline));
69 static inline ll rd() {
70     int c = getchar_();
71     while (__builtin_expect((c != '-') && (c < '0' || c > '9'), 1)) {
72         c = getchar_();
73         if (c == EOF) return 0;
74     }
75     int neg = 0;
76     if (c == '-') { neg = 1; c = getchar_(); }
77     ll x = 0;
78     while (c >= '0' && c <= '9') {
79         x = x * 10 + (c - '0');
80         c = getchar_();
81     }
82 }
```

```

81     return neg ? -x : x;
82 }
83
84 static const size_t OUT_BUF_SZ = 1 << 20;
85 static char outbuf[OUT_BUF_SZ];
86 static size_t out_pos = 0;
87
88 static inline void flush() {
89     if (out_pos) {
90         fwrite(outbuf, 1, out_pos, stdout);
91         out_pos = 0;
92     }
93 }
94
95 static inline void pt(char c) {
96     if (out_pos + 1 >= OUT_BUF_SZ) flush();
97     outbuf[out_pos++] = c;
98 }
99
100 static inline void pt(ll x) {
101     if (out_pos + 80 >= OUT_BUF_SZ) flush();
102     if (x == 0) {
103         outbuf[out_pos++] = '0';
104         outbuf[out_pos++] = '\n';
105         return;
106     }
107     if (x < 0) {
108         outbuf[out_pos++] = '-';
109         x = -x;
110     }
111     char tmp[64];
112     int tp = 0;
113     while (x > 0) {
114         tmp[tp++] = char('0' + (int)(x % 10));
115         x /= 10;
116     }
117     for (int i = tp - 1; i >= 0; --i) outbuf[out_pos++] = tmp[i];
118     outbuf[out_pos++] = '\n';
119 }
120
121 int main() {
122     int t = (int)rd();
123     while (t--) {
124         ll l = rd();

```

```

125     ll r = rd();
126     ll k = rd();
127     if (k < 2) k = 2;
128     if (k > r - l + 1) { pt('0'); pt('\n'); continue; }
129     ll limit = r / k;
130     if (limit <= 0) { pt('0'); pt('\n'); continue; }
131
132     ll lm = (l > 0 ? l - 1 : 0);
133     ll ans = 0;
134
135     for (ll g = 1; g <= limit; ) {
136         ll u = r / g;
137         ll v = (lm == 0 ? 0 : (lm / g));
138         ll r1 = (u == 0 ? r : (r / u));
139         ll r2 = (v == 0 ? limit : (lm / v));
140         ll ng = r1 < r2 ? r1 : r2;
141         if (ng > limit) ng = limit;
142         if (ng < g) ng = g;
143
144         ll cnt = u - v;
145         if (cnt < k) {
146             break;
147         }
148         ll cand = ng * ng;
149         cand *= u + v;
150         cand *= (cnt - 1);
151         cand /= 2;
152         if (cand > ans) ans = cand;
153
154         g = ng + 1;
155     }
156
157     pt(ans);
158 }
159     flush();
160     return 0;
161 }
```