

1 Mathematics

1.1 Extended Euclidean

```

1 // ax + by = gcd(a,b)
2 pair<long long, long long> extgcd(long long
  a, long long b)
3 {
4     if (b == 0)
5         return {1, 0};
6     long long k = a / b;
7     pair<long long, long long> p = extgcd(b,
8     a - k * b);
9     //cout << p.first << " " << p.second <<
10    endl;
11    //cout << "商數(k)= " << k << endl <<
12    endl;
13    return {p.second, p.first - k * p.second
14    };
15 }
16 int main()
17 {
18     int a, b;
19     cin >> a >> b;
20     pair<long long, long long> xy = extgcd(a
21     , b); //(x0,y0)
22     cout << xy.first << " " << xy.second <<
23     endl;
24     cout << xy.first << " * " << a << " + "
25     << xy.second << " * " << b << endl;
26     return 0;
27 }
28 // ax + by = gcd(a,b) * r
29 // find |x|+|y| -> min*/
30 int main()
31 {
32     long long r, p, q; /*px+qy = r*/
33     int cases;
34     cin >> cases;
35     while (cases--)
36     {
37         cin >> r >> p >> q;
38         pair<long long, long long> xy =
39         extgcd(q, p); //(x0,y0)
40         long long ans = 0, tmp = 0;
41         double k, k1;
42         long long s, s1;
43         k = 1 - (double)(r * xy.first) / p;
44         s = round(k);
45         ans = llabs(r * xy.first + s * p) +
46         llabs(r * xy.second - s * q);
47         k1 = -(double)(r * xy.first) / p;
48         s1 = round(k1);
49         /*cout << k << endl << k1 << endl;
50         cout << s << endl << s1 << endl;
51         */
52         tmp = llabs(r * xy.first + s1 * p) +
53         llabs(r * xy.second - s1 * q);
54         ans = min(ans, tmp);
55         cout << ans << endl;
56     }
57 }

```

```

48 return 0;
49 }

```

1.2 Hex to Dec

```

1 int HextoDec(string num) //16 to 10
2 {
3     int base = 1;
4     int temp = 0;
5     for (int i = num.length() - 1; i = 0; i
6     --)
7     {
8         if (num[i] == '0' && num[i] == '9')
9         {
10             temp += (num[i] - 48) base;
11             base = base 16;
12         }
13         else if (num[i] == 'A' && num[i] == 'F'
14         && num[i] == 'a' && num[i] == 'f')
15         {
16             temp += (num[i] - 55) base;
17             base = base 16;
18         }
19     }
20     return temp;
21 }
22 void DecToHex(int p_intValue) //10 to 16
23 {
24     char l_pCharRes = new (char);
25     sprintf(l_pCharRes, %X, p_intValue);
26     int l_intResult = stoi(l_pCharRes);
27     cout << l_pCharRes << endl;
28     return l_intResult;
29 }

```

1.3 PI

```

1 #define PI acos(-1)
2 #define PI M_PI

```

1.4 Prime table

```

1 // 埃拉托斯特尼篩法
2 const int maxn = 1000000;
3 bitset<maxn> prime;
4 void sieve()
5 {
6     for (int i = 2; i * i < maxn; ++i)
7     {
8         if (prime[i] == 0)
9         {
10             for (int j = i * i; j < maxn; j
11             += i)
12                 prime[j] = 1;
13         }
14     }
15 }

```

```

14 }
15 /* 0跟1要寫if過濾掉 */
16 // if(!prime[數字])
17 // 我是質數

```

1.5 二分逼近法

```

1 #define eps 1e-14
2 void half_interval()
3 {
4     double L = 0, R = /*區間*/, M;
5     while (R - L >= eps)
6     {
7         M = (R + L) / 2;
8         if (/*函數*/ > /*方程式目標*/)
9             L = M;
10        else
11            R = M;
12    }
13    printf("%.3lf\n", R);
14 }

```

1.6 四則運算

```

1 string s = "";
2 long long int DFS(int le, int ri) // (0,
3 string final index)
4 {
5     int c = 0;
6     for (int i = ri; i >= le; i--)
7     {
8         if (s[i] == ')')
9             c++;
10        if (s[i] == '(')
11            c--;
12        if (s[i] == '+' && c == 0)
13            return DFS(le, i - 1) + DFS(i +
14            1, ri);
15        if (s[i] == '-' && c == 0)
16            return DFS(le, i - 1) - DFS(i +
17            1, ri);
18    }
19    for (int i = ri; i >= le; i--)
20    {
21        if (s[i] == ')')
22            c++;
23        if (s[i] == '(')
24            c--;
25        if (s[i] == '*' && c == 0)
26            return DFS(le, i - 1) * DFS(i +
27            1, ri);
28        if (s[i] == '/' && c == 0)
29            return DFS(le, i - 1) / DFS(i +
30            1, ri);
31        if (s[i] == '%' && c == 0)
32            return DFS(le, i - 1) % DFS(i +
33            1, ri);
34    }
35    if ((s[le] == '(' && (s[ri] == ')'))

```

```

30 return DFS(le + 1, ri - 1); //去除刮
31 號
32 if (s[le] == ' ' && s[ri] == ' ')
33     return DFS(le + 1, ri - 1); //去除左
34     右兩邊空格
35 if (s[le] == ' ')
36     return DFS(le + 1, ri); //去除左邊空
37     格
38 if (s[ri] == ' ')
39     return DFS(le, ri - 1); //去除右邊空
40     格
41 long long int num = 0;
42 for (int i = le; i <= ri; i++)
43     num = num * 10 + s[i] - '0';
44 return num;
45 }

```

1.7 數字乘法組合

```

1 void toans(vector<vector<int>> &ans, vector<
2 int> com)
3 {
4     // sort(com.begin(), com.end());
5     ans.push_back(com);
6     // for (auto i : com)
7     //     cout << i << ' ';
8     // cout << endl;
9 }
10 void finds(int j, int old, int num, vector<
11 int> com, vector<vector<int>> &ans)
12 {
13     for (int i = j; i <= sqrt(num); i++)
14     {
15         if (old == num)
16             com.clear();
17         if (num % i == 0)
18         {
19             vector<int> a;
20             a = com;
21             a.push_back(i);
22             finds(i, old, num / i, a, ans);
23             a.push_back(num / i);
24             toans(ans, a);
25         }
26     }
27 }
28 int main()
29 {
30     vector<vector<int>> ans;
31     vector<int> zero;
32     finds(2, num, num, zero, ans);
33     // num 為 input 數字
34     for (int i = 0; i < ans.size(); i++)
35     {
36         for (int j = 0; j < ans[i].size() -
37         1; j++)
38             cout << ans[i][j] << " ";
39         cout << ans[i][ans[i].size() - 1] <<
40         endl;
41     }
42 }

```

1.8 數字加法組合

```

1 void printCombination(vector<int> const &out
  , int m, vector<vector<int>> &ans)
2 {
3     for (int i : out)
4         if (i > m)
5             return;
6     ans.push_back(out);
7 }
8
9 void recur(int i, int n, int m, vector<int>
  &out, vector<vector<int>> &ans)
10 {
11     if (n == 0)
12         printCombination(out, m, ans);
13     for (int j = i; j <= n; j++)
14     {
15         out.push_back(j);
16         recur(j, n - j, m, out, ans);
17         out.pop_back();
18     }
19 }
20 int main()
21 {
22     vector<vector<int>> ans;
23     vector<int> zero;
24     recur(1, num, num, zero, ans);
25     // num 為 input 數字
26     for (int i = 0; i < ans.size(); i++)
27     {
28         for (int j = 0; j < ans[i].size() -
29             1; j++)
30             cout << ans[i][j] << " ";
31         cout << ans[i][ans[i].size() - 1] <<
32             endl;
33     }
34 }

```

1.9 羅馬數字

```

1 int romanToInt(string s)
2 {
3     unordered_map<char, int> T;
4     T['I'] = 1;
5     T['V'] = 5;
6     T['X'] = 10;
7     T['L'] = 50;
8     T['C'] = 100;
9     T['D'] = 500;
10    T['M'] = 1000;
11
12    int sum = T[s.back()];
13    for (int i = s.length() - 2; i >= 0; --i)
14    {
15        if (T[s[i]] < T[s[i + 1]])
16            sum -= T[s[i]];
17        else
18            sum += T[s[i]];
19    }
20    return sum;
21 }

```

21 }

1.10 質因數分解

```

1 void cal(int in)
2 {
3     for (long long x = 2; x <= in; x++)
4     {
5         while (in % x == 0)
6         {
7             cout << x << " ";
8             in /= x;
9         }
10    }
11 }

```

2 data structure

2.1 BigInt

```

1 //台大
2 struct BigInt{
3     static const int LEN = 60;
4     static const int BIGMOD = 10000;
5     int s;
6     int vl, v[LEN];
7     // vector<int> v;
8     BigInt() : s(1) { vl = 0; }
9     BigInt(long long a) {
10         s = 1; vl = 0;
11         if (a < 0) { s = -1; a = -a; }
12         while (a) {
13             push_back(a % BIGMOD);
14             a /= BIGMOD;
15         }
16     }
17     BigInt(string str) {
18         s = 1; vl = 0;
19         int stPos = 0, num = 0;
20         if (!str.empty() && str[0] == '-') {
21             stPos = 1;
22             s = -1;
23         }
24         for (int i = SZ(str)-1, q=1; i>=stPos; i--) {
25             num += (str[i] - '0') * q;
26             if ((q *= 10) >= BIGMOD) {
27                 push_back(num);
28                 num = 0; q = 1;
29             }
30         }
31         if (num) push_back(num);
32         n();
33     }
34     int len() const {
35         return vl; //return SZ(v);
36     }
37 }

```

```

37 bool empty() const { return len() == 0; }
38 void push_back(int x) {
39     v[vl++] = x; //v.PB(x);
40 }
41 void pop_back() {
42     vl--; //v.pop_back();
43 }
44 int back() const {
45     return v[vl-1]; //return v.back();
46 }
47 void n() {
48     while (!empty() && !back()) pop_back();
49 }
50 void resize(int nl) {
51     vl = nl; //v.resize(nl);
52     fill(v, v+vl, 0); //fill(ALL(v), 0);
53 }
54 void print() const {
55     if (empty()) { putchar('0'); return; }
56     if (s == -1) putchar('-');
57     printf("%d", back());
58     for (int i=len()-2; i>=0; i--)
59         printf("%.4d", v[i]);
60 }
61 friend std::ostream& operator << (std::
62     ostream& out, const BigInt &a) {
63     if (a.empty()) { out << "0"; return
64         out; }
65     if (a.s == -1) out << "-";
66     out << a.back();
67     for (int i=a.len()-2; i>=0; i--) {
68         char str[10];
69         snprintf(str, 5, "%.4d", a.v[i]);
70         out << str;
71     }
72     return out;
73 }
74 int cp3(const BigInt &b) const {
75     if (s != b.s) return s - b.s;
76     if (s == -1) return -(*this).cp3(-b);
77     if (len() != b.len()) return len()-b
78         .len(); //int
79     for (int i=len()-1; i>=0; i--)
80         if (v[i] != b.v[i]) return v[i]-b
81             .v[i];
82     return 0;
83 }
84 bool operator<(const BigInt &b) const
85 { return cp3(b)<0; }
86 bool operator<=(const BigInt &b) const
87 { return cp3(b)<=0; }
88 bool operator==(const BigInt &b) const
89 { return cp3(b)==0; }
90 bool operator!=(const BigInt &b) const
91 { return cp3(b)!=0; }
92 bool operator>(const BigInt &b) const
93 { return cp3(b)>0; }
94 bool operator>=(const BigInt &b) const
95 { return cp3(b)>=0; }
96 BigInt operator - () const {
97     BigInt r = (*this);
98     r.s = -r.s;
99     return r;
100 }
101 BigInt operator + (const BigInt &b)
102 const {
103     if (s == -1) return -(*this)+(-b);
104     ;
105     if (b.s == -1) return (*this)-(-b);
106     BigInt r;
107     int nl = max(len(), b.len());
108     r.resize(nl + 1);
109     for (int i=0; i<nl; i++) {
110         if (i < len()) r.v[i] += v[i];
111         if (i < b.len()) r.v[i] += b.v[i];
112         if (r.v[i] >= BIGMOD) {
113             r.v[i+1] += r.v[i] / BIGMOD;
114             r.v[i] %= BIGMOD;
115         }
116     }
117     r.n();
118     return r;
119 }
120 BigInt operator - (const BigInt &b)
121 const {
122     if (s == -1) return -(*this)-(-b);
123     ;
124     if (b.s == -1) return (*this)+(-b);
125     if ((*this) < b) return -(b-(*this))
126     ;
127     BigInt r;
128     r.resize(len());
129     for (int i=0; i<len(); i++) {
130         r.v[i] += v[i];
131         if (i < b.len()) r.v[i] -= b.v[i];
132         if (r.v[i] < 0) {
133             r.v[i] += BIGMOD;
134             r.v[i+1]--;
135         }
136     }
137     r.n();
138     return r;
139 }
140 BigInt operator * (const BigInt &b) {
141     BigInt r;
142     r.resize(len() + b.len() + 1);
143     r.s = s * b.s;
144     for (int i=0; i<len(); i++) {
145         for (int j=0; j<b.len(); j++) {
146             r.v[i+j] += v[i] * b.v[j];
147             if (r.v[i+j] >= BIGMOD) {
148                 r.v[i+j+1] += r.v[i+j] /
149                     BIGMOD;
150                 r.v[i+j] %= BIGMOD;
151             }
152         }
153     }
154     r.n();
155     return r;
156 }
157 BigInt operator / (const BigInt &b) {
158     BigInt r;
159     r.resize(max(1, len()-b.len()+1));
160     int oriS = s;
161     BigInt b2 = b; // b2 = abs(b)
162 }

```

```

93 r.s = -r.s;
94 return r;
95 }
96 BigInt operator + (const BigInt &b)
97 const {
98     if (s == -1) return -(*this)+(-b);
99     ;
100     if (b.s == -1) return (*this)-(-b);
101     BigInt r;
102     int nl = max(len(), b.len());
103     r.resize(nl + 1);
104     for (int i=0; i<nl; i++) {
105         if (i < len()) r.v[i] += v[i];
106         if (i < b.len()) r.v[i] += b.v[i];
107         if (r.v[i] >= BIGMOD) {
108             r.v[i+1] += r.v[i] / BIGMOD;
109             r.v[i] %= BIGMOD;
110         }
111     }
112     r.n();
113     return r;
114 }
115 BigInt operator - (const BigInt &b)
116 const {
117     if (s == -1) return -(*this)-(-b);
118     ;
119     if (b.s == -1) return (*this)+(-b);
120     if ((*this) < b) return -(b-(*this))
121     ;
122     BigInt r;
123     r.resize(len());
124     for (int i=0; i<len(); i++) {
125         r.v[i] += v[i];
126         if (i < b.len()) r.v[i] -= b.v[i];
127         if (r.v[i] < 0) {
128             r.v[i] += BIGMOD;
129             r.v[i+1]--;
130         }
131     }
132     r.n();
133     return r;
134 }
135 BigInt operator * (const BigInt &b) {
136     BigInt r;
137     r.resize(len() + b.len() + 1);
138     r.s = s * b.s;
139     for (int i=0; i<len(); i++) {
140         for (int j=0; j<b.len(); j++) {
141             r.v[i+j] += v[i] * b.v[j];
142             if (r.v[i+j] >= BIGMOD) {
143                 r.v[i+j+1] += r.v[i+j] /
144                     BIGMOD;
145                 r.v[i+j] %= BIGMOD;
146             }
147         }
148     }
149     r.n();
150     return r;
151 }
152 BigInt operator / (const BigInt &b) {
153     BigInt r;
154     r.resize(max(1, len()-b.len()+1));
155     int oriS = s;
156     BigInt b2 = b; // b2 = abs(b)
157 }

```

```

151 s = b2.s = r.s = 1;
152 for (int i=r.len()-1; i>=0; i--) {
153     int d=0, u=BIGMOD-1;
154     while(d<u) {
155         int m = (d+u+1)>>1;
156         r.v[i] = m;
157         if((r*b2) > (*this)) u = m
158             -1;
159         else d = m;
160     }
161     r.v[i] = d;
162 }
163 s = oriS;
164 r.s = s * b.s;
165 r.n();
166 return r;
167 }
168 Bigint operator % (const Bigint &b) {
169     return (*this)-(*this)/b*b;
170 }
171 };

```

2.2 分數

```

1 class Rational
2 {
3     friend istream &operator>>(istream &,
4         Rational &);
5     friend ostream &operator<<(ostream &,
6         const Rational &);
7 public:
8     Rational() //constructor one
9     {
10         m_numerator = 0;
11         m_denominator = 1;
12     }
13     Rational(int a, int b) //constructor two
14     {
15         if (b < 0 || b == 0) //avoids negative
16             denominators. && prevents a 0
17             denominator
18         {
19             cout << "This Rational number can't be
20                 used.\n\n";
21             m_numerator = 0;
22             m_denominator = 0;
23         }
24         else
25         {
26             cout << "This Rational number can be
27                 used.\n\n";
28             m_numerator = a;
29             m_denominator = b;
30         }
31     }
32     Rational operator+(const Rational& a); //
33     Rational operator-(const Rational& a); //
34     Rational operator*(const Rational& a); //
35     Rational operator/(const Rational& a); //

```

```

29 Rational operator/(const Rational& a); //
30     除
31     bool operator==(const Rational& a); //相
32     等
33     void reduce(); //化簡
34 private:
35     int m_numerator;
36     int m_denominator;
37 };
38 istream &operator>>(istream &input, Rational
39     &test )
40 {
41     char temp;
42     input >> test.m_numerator;
43     input >> temp;
44     input >> test.m_denominator;
45     Rational final(test.m_numerator, test.
46         m_denominator); //final用來告訴使用者
47     這數字不符合!
48     if (test.m_denominator < 0 || test.
49         m_denominator == 0) //不符合(再輸入
50         一次)
51     {
52         while (test.m_denominator < 0 || test.
53             m_denominator == 0) //有可能輸入的
54             東西還是不符合,所以用迴圈
55     {
56         cout << "Enter another Rational number
57             (n/d): ";
58         input >> test.m_numerator;
59         input >> temp;
60         input >> test.m_denominator;
61         Rational final(test.m_numerator, test.
62             m_denominator); //final用來告訴使
63             用者這數字不符合!
64     }
65     return input;
66 }
67 else
68     return input;
69 }
70 ostream &operator<<(ostream &output, const
71     Rational &test )
72 {
73     output << test.m_numerator;
74     if(test.m_denominator == 0)
75         return output;
76     if (test.m_denominator == 1)
77         return output;
78     else
79     {
80         output << "/";
81         output << test.m_denominator;
82     }
83     return output;
84 }
85 Rational Rational::operator+(const Rational&
86     a)
87 {
88     Rational c;
89     c.m_denominator = this->m_denominator * a.
90         m_denominator; //通分(同乘)
91 }

```

```

77 c.m_numerator = (this->m_numerator * a.
78     m_denominator) + (a.m_numerator * this
79     ->m_denominator);
80 c.reduce();
81 return c;
82 }
83 Rational Rational::operator-(const Rational&
84     a)
85 {
86     Rational c;
87     c.m_denominator = this->m_denominator * a.
88         m_denominator;
89     c.m_numerator = (this->m_numerator * a.
90         m_denominator) - (a.m_numerator * this
91         ->m_denominator);
92 c.reduce();
93 return c;
94 }
95 Rational Rational::operator*(const Rational&
96     a)
97 {
98     Rational c;
99     c.m_denominator = this->m_denominator * a.
100         m_denominator;
101     c.m_numerator = this->m_numerator * a.
102         m_numerator;
103 c.reduce();
104 return c;
105 }
106 Rational Rational::operator/(const Rational&
107     a)
108 {
109     Rational c;
110     c.m_denominator = this->m_denominator * a.
111         m_numerator;
112     c.m_numerator = this->m_numerator * a.
113         m_denominator;
114 c.reduce();
115 return c;
116 }
117 bool Rational::operator==(const Rational& a)
118 {
119     if (m_numerator == a.m_numerator)
120     {
121         if (m_denominator == a.m_denominator)
122             return true;
123         else
124             return false;
125     }
126     else
127         return false;
128 }
129 void Rational::reduce()
130 {
131     int i;
132     int max;
133     if(m_numerator > m_denominator)
134         max = m_numerator;
135     else
136         max = m_denominator;
137     for (i = 2; i <= max; i++)
138     {
139         if (m_denominator % i == 0 && m_numerator
140             % i == 0)
141         {
142             m_denominator /= i;
143             m_numerator /= i;
144         }
145     }
146 }

```

```

130 m_numerator /= i;
131 i = 1;
132 max = m_denominator;
133 continue;
134 }
135 }

```

ACM ICPC
TEAM
REFERENCE -

ANGRY CROW
TAKES FLIGHT!

Contents

1	Mathematics	1	1.9	羅馬數字	2
1.1	Extended Euclidean	1	1.10	質因數分解	2
1.2	Hex to Dec	1			
1.3	PI	1	2	data structure	2
1.4	Prime table	1	2.1	Bigint	2
1.5	二分逼近法	1	2.2	分數	3