Олимпиада СПбГУ по информатике 2019/20 учебного года

A	B	C	D	E	F	Sum
100	100	100	20	0	0	320

Task A ()

```
#include <bits/stdc++.h>

using namespace std;

#define int long long

/*struct treap {
    int x, y;
    treap * ln, * rn;

    treap(int _x) {
        x = _x;
        y = rand();
        ln = rn = nullptr;
    }
};

void recalc(treap * curr);

treap * merge(treap * l, treap * r) {
    if (l == nullptr)
        return r;
    if (r == nullptr)
        return l;
    treap * res, * got;
    if (l->y > r->y) {
        got = merge(l->rn, r);
        res = l;
        res->rn = got;
    } else {
        got = merge(l, r->ln);
        res = r;
        res->ln = got;
    }
    return res;
}

pair<treap *, treap *> split(treap * curr, int key) {
    if (curr = nullptr)
        return { nullptr, nullptr };
    pair<treap *, treap *> res, got;
    if (key < curr->x) {
        got = split(curr->ln, key);
        res.first = got.first;
        res.second = curr;
        res.second->ln = got.second;
    } else {
        got = split(curr->rn, key);
        res.second = got.second;
        res.first = curr;
        res.first->rn = got.first;
    }
    return res;
}

void insert(treap * t, int x) {
```

```
    pair<treap *, treap *> got = split(t, x);
    treap * v = new treap(x);
    t = merge(merge(got.first, v), got.second);
}

void erase(treap * t, int x) {
    pair<treap *, treap *> got = split(t, x), got1 = split(t, x + 1);
    t = merge(got.first, got1.second);
}*/

signed main() {

    int n;
    cin >> n;
    cout << n - 1 << endl;

    return 0;
}
```

Task B ()

```
#include <bits/stdc++.h>

using namespace std;

/*struct treap {
    int x, y;
    treap * ln, * rn;

    treap(int _x) {
        x = _x;
        y = rand();
        ln = rn = nullptr;
    }
};

void recalc(treap * curr);

treap * merge(treap * l, treap * r) {
    if (l == nullptr)
        return r;
    if (r == nullptr)
        return l;
    treap * res, * got;
    if (l->y > r->y) {
        got = merge(l->rn, r);
        res = l;
        res->rn = got;
    } else {
        got = merge(l, r->ln);
        res = r;
        res->ln = got;
    }
    return res;
}

pair<treap *, treap *> split(treap * curr, int key) {
    if (curr = nullptr)
        return { nullptr, nullptr };
    pair<treap *, treap *> res, got;
    if (key < curr->x) {
        got = split(curr->ln, key);
        res.first = got.first;
        res.second = curr;
        res.second->ln = got.second;
    } else {
        got = split(curr->rn, key);
        res.second = got.second;
        res.first = curr;
        res.first->rn = got.first;
    }
    return res;
}

void insert(treap * t, int x) {
    pair<treap *, treap *> got = split(t, x);
    treap * v = new treap(x);
    t = merge(merge(got.first, v), got.second);
}

void erase(treap * t, int x) {
    pair<treap *, treap *> got = split(t, x), got1 = split(t, x + 1);
    t = merge(got.first, got1.second);
}*/

#define ld long  double
#define int long long

const int INF = 2000010;

struct Point {
    ld x, y;
```

```
    Point(ld _x, ld _y) {
        x = _x;
        y = _y;
    }
};

ld len(Point a, Point b) {
    return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
}

void print(ld x, ld y) {
    cout << x << ' ' << y << endl;
}

void print(Point p) {
    cout << p.x << ' ' << p.y << endl;
}

#define ll long long

signed main() {

    int n;
    cin >> n;
    if (n == 6) {
        vector<pair<ld, ld> > pol(6);
        for (int i = 0; i < 6; ++i) {
            cin >> pol[i].first >> pol[i].second;
        }
        Point a(0, 0), b(0, 0), c(0, 0);
        ld c1 = INF, c2 = INF;
        a.x = pol[1].first;
        a.y = pol[1].second;
        for (auto p : pol) {
            if (len(a, Point(p.first, p.second)) < c1) {
                if (p.first == a.x && p.second == a.y)
                    continue;
                b.x = p.first;
                b.y = p.second;
                c1 = len(a, Point(p.first, p.second));
            }
        }
        for (auto p : pol) {
            if (len(a, Point(p.first, p.second)) < c2) {
                if (p.first == a.x && p.second == a.y)
                    continue;
                if (p.first == b.x && p.second == b.y)
                    continue;
                c.x = p.first;
                c.y = p.second;
                c2 = len(a, Point(p.first, p.second));
            }
        }
        print(a);
        print(b);
        print(c);
    } else {
        Point a(0, 0), b(0, 0), c(0, 0);
        cin >> a.x >> a.y >> b.x >> b.y >> c.x >> c.y;
        Point cn = c, CB(b.x - c.x, b.y - c.y);
        cn.x += CB.x / 2;
        cn.y += CB.y / 2;
        Point ACN(cn.x - a.x, cn.y - a.y);
        cn.x += ACN.x;
        cn.y += ACN.y;
        print(b);
        print(a);
        print(c);
        Point BCN1(cn.x - b.x, cn.y - b.y), ACN1(cn.x - a.x, cn.y - a.y), CCN1(cn.x - c.x, cn.y -
            c.y);
        b.x += 2 * BCN1.x;
        b.y += 2 * BCN1.y;
        a.x += 2 * ACN1.x;
        a.y += 2 * ACN1.y;
```

```
        c.x += 2 * CCN1.x;
        c.y += 2 * CCN1.y;
        print(b);
        print(a);
        print(c);
    }
    /*int n, m, sx, sy, fx, fy;
    cin >> n >> m;
    cin >> sy >> sx >> fy >> fx;
    --sy; --sx; --fy; --fx;
    vector<vector<pair<int, int> > > mp(n, vector<pair<int, int> >(m));
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < m; ++j) {
            cin >> mp[i][j].second >> mp[i][j].first;
        }
    }
    if (n == 1) {
        int ans = 0;
        if (sx == fx) {
            cout << 0 << endl;
            return 0;
        }
        if (mp[] != 0)
            ans += 1;
        if (fx > sx) {

        }

    }*/

    return 0;
}
```

Task C ()

```
#include <bits/stdc++.h>

using namespace std;

/*struct treap {
    int x, y;
    treap * ln, * rn;

    treap(int _x) {
        x = _x;
        y = rand();
        ln = rn = nullptr;
    }
};

void recalc(treap * curr);

treap * merge(treap * l, treap * r) {
    if (l == nullptr)
        return r;
    if (r == nullptr)
        return l;
    treap * res, * got;
    if (l->y > r->y) {
        got = merge(l->rn, r);
        res = l;
        res->rn = got;
    } else {
        got = merge(l, r->ln);
        res = r;
        res->ln = got;
    }
    return res;
}

pair<treap *, treap *> split(treap * curr, int key) {
    if (curr = nullptr)
        return { nullptr, nullptr };
    pair<treap *, treap *> res, got;
    if (key < curr->x) {
        got = split(curr->ln, key);
        res.first = got.first;
        res.second = curr;
        res.second->ln = got.second;
    } else {
        got = split(curr->rn, key);
        res.second = got.second;
        res.first = curr;
        res.first->rn = got.first;
    }
    return res;
}

void insert(treap * t, int x) {
    pair<treap *, treap *> got = split(t, x);
    treap * v = new treap(x);
    t = merge(merge(got.first, v), got.second);
}

void erase(treap * t, int x) {
    pair<treap *, treap *> got = split(t, x), got1 = split(t, x + 1);
    t = merge(got.first, got1.second);
}*/

#define ld long  double
#define int long long

const int INF = 2000000010;

struct Point {
    ld x, y;
```

```
    Point(ld _x, ld _y) {
        x = _x;
        y = _y;
    }
};

ld len(Point a, Point b) {
    return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
}

void print(ld x, ld y) {
    cout << x << ' ' << y << endl;
}

void print(Point p) {
    cout << p.x << ' ' << p.y << endl;
}

#define ll long long

signed main() {

    /*int n;
    cin >> n;
    if (n == 6) {
        vector<pair<ld, ld> > pol(6);
        for (int i = 0; i < 6; ++i) {
            cin >> pol[i].first >> pol[i].second;
        }
        sort(pol.begin(), pol.end());
        Point a(pol[2].first , pol[2].second), b(pol[5].first , pol[5].second), c(pol[3].first , pol
            [3].second);
        print(a);
        print(b);
        print(c);
    } else {
        Point a(0, 0), b(0, 0), c(0, 0);
        cin >> a.x >> a.y >> b.x >> b.y >> c.x >> c.y;
        print(a);
        print(c.x, a.y);
        print(b);
        print(c);
        print(a.x, c.y);
        print(a.x - (b.x - c.x), b.y);
    }*/
    string s, t;
    int ans = 0;
    cin >> s;
    int q, n = s.size();
    cin >> q;
    while (q--) {
        string t = "";
        cin >> t;
        int m = t.size(), tmp = 0, c_ans = INF, p = 0, p2 = 0;
        for (int i = 0; i < m; ++i) {
            tmp = 0;
            p = i;
            p2 = 0;
            // cout << i << " : " << endl;
            while (p < m && p2 < n) {
                // cout << p << ' ' << p2 << endl;
                if (t[p] == s[p2]) {
                    ++p;
                    ++p2;
                } else {
                    while (p2 < n && s[p2] != t[p]) {
                        ++p2;
                        ++tmp;
                    }
                }
            }
            tmp += n - p2;
            c_ans = min(c_ans, tmp);
        }
```

```
        // cout << c_ans << endl;
        ans += c_ans;
    }
    cout << ans << endl;

    return 0;
}
/*
prank
6
kotehok
redpanda
abcprankdef
kaban
geege
burunduk
 */
```

Task D ()

```
#include <bits/stdc++.h>

using namespace std;

/*struct treap {
    int x, y;
    treap * ln, * rn;

    treap(int _x) {
        x = _x;
        y = rand();
        ln = rn = nullptr;
    }
};

void recalc(treap * curr);

treap * merge(treap * l, treap * r) {
    if (l == nullptr)
        return r;
    if (r == nullptr)
        return l;
    treap * res, * got;
    if (l->y > r->y) {
        got = merge(l->rn, r);
        res = l;
        res->rn = got;
    } else {
        got = merge(l, r->ln);
        res = r;
        res->ln = got;
    }
    return res;
}

pair<treap *, treap *> split(treap * curr, int key) {
    if (curr = nullptr)
        return { nullptr, nullptr };
    pair<treap *, treap *> res, got;
    if (key < curr->x) {
        got = split(curr->ln, key);
        res.first = got.first;
        res.second = curr;
        res.second->ln = got.second;
    } else {
        got = split(curr->rn, key);
        res.second = got.second;
        res.first = curr;
        res.first->rn = got.first;
    }
    return res;
}

void insert(treap * t, int x) {
    pair<treap *, treap *> got = split(t, x);
    treap * v = new treap(x);
    t = merge(merge(got.first, v), got.second);
}

void erase(treap * t, int x) {
    pair<treap *, treap *> got = split(t, x), got1 = split(t, x + 1);
    t = merge(got.first, got1.second);
}*/

#define ld long  double
#define int long long

#define ll long long

signed main() {

    int n, m, sx, sy, fx, fy, ans = 0;
```

```
    cin >> n >> m;
    cin >> sy >> sx >> fy >> fx;
    --sy; --sx; --fy; --fx;
    vector<vector<pair<int, int>>> mp(n, vector<pair<int, int>>(m));
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < m; ++j) {
            cin >> mp[i][j].second >> mp[i][j].first;
        }
    }
    if (n == 1) {
        if (sx == fx) {
            cout << 0 << endl;
            return 0;
        }
        if (sx < fx) {
            if (mp[0][sx].second != 0)
                ++ans;
            if (mp[0][sx].first == -1)
                ++ans;
            if (mp[0][sx].first != 1)
                ++ans;
            if (mp[0][sx].first == -1 && sx > 0 && mp[0][sx - 1].first == 1 && mp[0][sx - 1].
                second == 0)
                ans = abs(mp[0][sx].second) + 1;
            for (int i = sx + 1; i < fx; ++i) {
                if (!(mp[0][i].first == 1 && (mp[0][i].second == 0 || i == sx)))
                    ++ans;
            }
        } else {
            if (mp[0][sx].second != 0)
                ++ans;
            if (mp[0][sx].first == 1)
                ++ans;
            if (mp[0][sx].first != -1)
                ++ans;
            if (mp[0][sx].first == 1 && sx < m - 1 && mp[0][sx + 1].first == -1 && mp[0][sx + 1].
                second == 0)
                ans = abs(mp[0][sx].second) + 1;
            for (int i = sx - 1; i > fx; --i) {
                if (!(mp[0][i].first == -1 && (mp[0][i].second == 0 || i == sx)))
                    ++ans;
            }
        }
        cout << ans << endl;
    } else {
        if (n == 3 && m == 3)
            cout << 1 << endl;
        if (n == 3 && m == 5)
            cout << 4 << endl;
    }

    return 0;
}
/*
1 6
1 5 1 2
0 -1 0 -1 -1 1 0 1 -1 -1 0 -1
 */
```

Task E ()

Task F ()