Олимпиада СПбГУ по информатике 2018/19 учебного года

Зайнуллин Валерий Владиславович

A	B	C	D	E	F	Sum
100	100	100	60	100	7	467

Task A (100)

```cpp
#include <cstdio>
#include <vector>
#include <algorithm>

int main() {
        int n, m;
        scanf("%d %d", &n, &m);
    if (m % n != 0) {
        printf("No\n");
    } else {
        int nA = m / n;
        if ((nA & (nA - 1)) == 0) {
            printf("Yes\n");
        } else {
            printf("No\n");
        }
    }
        return 0;
}
```

Task B (100)

```cpp
#include <cstdio>
#include <vector>

int main() {
    int len;
    scanf("%d", &len);
    std::vector<char> str(len);
    for (char& ch: str) {
        scanf(" %c", &ch);
    }
    for (int i = 0; i < len - 1; ++i) {
        if ((str[i] == 'o' && str[i + 1] == 'r') || (str[i] == 'r' && str[i + 1] == 'o')) {
            printf("Yes\n");
            return 0;
        }
    }
    for (int i = 0; i < len - 2; ++i) {
        if (str[i] == 'o' && str[i + 2] == 'r') {
            printf("Yes\n");
            return 0;
        }
    }
    printf("No\n");
    return 0;
}
```

Task C (100)

```cpp
#include <cstdio>
#include <vector>
#include <functional>
#include <algorithm>

int main() {
    int nV;
    scanf("%d", &nV);
    std::vector<std::vector<int>> adj(nV);
    for (int iE = 0; iE < nV - 1; ++iE) {
        int from, to;
        scanf("%d %d", &from, &to);
        from -= 1;
        to -= 1;
        adj[from].push_back(to);
        adj[to].push_back(from);
    }
    std::vector<bool> visited(nV, false);
    std::vector<int> nVSub(nV, 1);
    std::vector<int> result(nV);
    std::function<void(const int)> dfs = [&](const int iV) {
        visited[iV] = true;
        int curRes = 0;
        for (int next: adj[iV]) {
            if (visited[next]) {
                continue;
            }
            dfs(next);
            curRes = std::max(curRes, nVSub[next]);
            nVSub[iV] += nVSub[next];
        }
        curRes = std::max(curRes, nV - nVSub[iV]);
        curRes += 1;
        result[iV] = curRes;
    };
    dfs(0);
    for (int v: result) {
        printf("%d ", v);
    }
    printf("\n");
    return 0;
}
```

Task D (60)

```
cmd = input()
t, n, p = map(int, input().split())
assert p == 7
if cmd == "split":
    for it in range(t):
        pwd = input()
        for im in range(n - 1):
            g = im & 1
            print(pwd[0+3*g:6+3*g] + ['a', 'b'][g], end='␣')
        print(pwd[:3] + pwd[6:9] + 'c')
else:
    for it in range(t):
        pwd = input().split()
        pwd.sort(key=lambda e: e[-1])
        if pwd[-1][-1] == 'c':
            if pwd[0][-1] == 'a':
                print(pwd[0][:6] + pwd[-1][3:6])
            else:
                print(pwd[-1][:3] + pwd[0][:6])
        else:
            print(pwd[0][:6] + pwd[-1][3:6])
```

Task E (100)

```cpp
#include <cstdio>
#include <vector>
#include <functional>
#include <algorithm>
#include <cassert>

#ifdef ONPC
#define __int128 long long
#endif

struct Point {
    long long x;
    long long y;
    int id;
};

int main() {
    int nP;
    scanf("%d", &nP);
    if (nP == 1) {
        printf("1\n");
        return 0;
    }
    std::vector<Point> pts(nP);
    for (int iP = 0; iP < nP; ++iP) {
        Point& pt = pts[iP];
        scanf("%lld %lld", &pt.x, &pt.y);
        pt.id = iP + 1;
    }
    long long x1, y1, x2, y2;
    scanf("%lld %lld %lld %lld", &x1, &y1, &x2, &y2);
    std::function<bool(const Point& left, const Point& right)> cmp;
    long long c;
    if (x1 == x2) {
        if (y1 > y2) {
            for (Point& pt: pts) {
                pt.y = 2 * y1 - pt.y;
            }
            y2 = 2 * y1 - y2;
        }
        assert(y1 <= y2);
        c = -x1 * (y2 - y1) + y1 * (x2 - x1);
        cmp = [&](const Point& left, const Point& right) {
            if (left.y != right.y) {
                return left.y < right.y;
            }
            return std::abs(x1 - left.x) > std::abs(x1 - right.x);
        };
    } else {
        if (y1 > y2) {
            for (Point& pt: pts) {
                pt.y = 2 * y1 - pt.y;
            }
            y2 = 2 * y1 - y2;
        }
        if (x1 > x2) {
            for (Point& pt: pts) {
                pt.x = 2 * x1 - pt.x;
            }
            x2 = 2 * x1 - x2;
        }
        assert(y1 <= y2);
        assert(x1 <= x2);
        c = -x1 * (y2 - y1) + y1 * (x2 - x1);
        cmp = [&](const Point& left, const Point& right) {
            __int128 val = ((__int128)-2*(y2 - y1)*left.y-2*(x2 - x1) * left.x) - ((__int128)-2*(
                y2 - y1)*right.y-2*(x2 - x1) *right.x);
            //printf("left.id = %d right.id = %d val = %lld", left.id, right.id, val);
            if (val != 0) {
```

```
                return val > 0;
            }
            return (__int128) c * (- 2 * left.y + 2 * right.y) > (__int128) (x2 - x1) * (-left.x *
                 left.x -left.y * left.y + right.y * right.y + right.x * right.x);
        };
    }
    std::sort(pts.begin(), pts.end(), cmp);
    for (Point& pt: pts) {
        //printf("x = %lld y = %lld\n", pt.x, pt.y);
    }
    if (cmp(pts[nP - 2], pts[nP - 1])) {
        printf("%d\n", pts.back().id);
    } else {
        printf("-1");
    }
    return 0;
}
```

Task F (7)

```cpp
#include <cstdio>
#include <vector>
#include <algorithm>

int main() {
    int nD, nP;
    scanf("%d %d", &nD, &nP);
    std::vector<int> d1(nD);
    std::vector<int> d2(nD);
    for (int i = 0; i < nD; ++i) {
        scanf("%d %d", &d1[i], &d2[i]);
    }
    int cc = 0;
    for (int mask = 0; mask < (1 << nD); ++mask) {
        int result = 0;
        int na = 0;
        int nb = 0;
        for (int d = 0; d < nD; ++d) {
            if (mask & (1 << d)) {
                na += nP;
            } else {
                nb += nP;
            }
            int ca = std::min(na, d1[d]);
            na -= ca;
            result += ca;
            int cb = std::min(nb, d2[d]);
            nb -= cb;
            result += cb;
        }
        cc = std::max(cc, result);
    }
    printf("%d\n", cc);
    return 0;
}
```