МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М.В. ЛОМОНОСОВА

ОЛИМПИАДНАЯ РАБОТА

Наименование олимпиады школьников: **«Ломоносов»**

Профиль олимпиады: **Информатика**

ФИО участника олимпиады: **Мачуговский Иван Александрович**

Класс: **10**

Технический балл: **100**

Дата проведения: **09 марта 2021 года**

Результаты проверки:

№	1	2	3	4	5	6	7
Оценка	20	20	20	10	20	15	15

Задача 1.

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



using namespace std;



int parse27(char c) {

	if('0' <= c && c <= '9') {

		return c - '0';

	} else if('A' <= c && c <= 'D') {

		return (c - 'A') + 10;

	} else {

		return -(('Z' - c) + 1);

	}

}



int main() {
```

```
char x;

cin >> x;



string s;

cin >> s;



reverse(s.begin(), s.end());

while(s.size() > 0 && s.back() == '0') {

	s.pop_back();

}

if(s.size() == 0) {

	s = "0";

}



map<int, string> map9;
```

```
	for(char c1: string("WXYZ1234")) {

		for(char c2: string("WXYZ01234")) {

			for(char c3: string("WXYZ01234")) {

				map9[parse27(c1) * 81 + parse27(c2) * 9 + parse27(c3)] = string(1, c1) + string(1, c2) + string(1, c2);

			}

			map9[parse27(c1) * 9 + parse27(c2)] = string(1, c1) + string(1, c2);

		}

		map9[parse27(c1)] = string(1, c1);

	}

	map9[0] = "0";



	int ans = 0;



	for(int i = 0; i < s.size(); i += 2) {

		string t = s.substr(i, 2);
```

```
            int n = parse27(t[0]) + (t.size() == 1 ? 0 : parse27(t[1]) * 27);
            string mapped = map9[n];
            if(i + 2 < s.size()) {
                mapped = string(3 - mapped.size(), '0') + mapped;
            }
            for(char c: mapped) {
                if(c == x) {
                    ans++;
                }
            }
        }


        cout << ans << endl;


        return 0;
}
```

Задача 2.

```
n = int(input())

f = list(map(int, input().split()))




def s9(x):
    if x == 0:
        return "0"
    s = ""
    while x != 0:
        digit = x % 9
        if digit >= 5:
            digit -= 9
        s += "01234WXYZ"[digit]
        x = (x - digit) // 9
    return s[::-1]
```

```
s9f = [s9(x) for x in f]



vampires = set()

for i in range(n):

    for j in range(i):

        a, b = f[i], f[j]

        s9a, s9b = s9f[i], s9f[j]

        vampire = a * b

        if not (a % 9 == b % 9 == 0) and len(s9a) * 2 == len(s9b) * 2 ==
len(s9(vampire)) and sorted(s9(vampire)) == sorted(s9a + s9b):

            vampires.add(vampire)



print(len(vampires))
```

Задача 3.

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

using namespace std;

int n, m, k;

vector< pair<int, int> > posof;

struct Tick {
	int vx;
	int vy;
	int l;
};

Tick vv[5001];
```

```
int main() {
	cin >> n >> m >> k;
	posof.resize(k + 1, make_pair(-1, -1));

	vector< pair<int, int> > events;
	for(int t = 0; t < k; t++) {
		int s, f;
		cin >> s >> f;
		events.emplace_back(s, t + 1);
		events.emplace_back(f, -(t + 1));
	}
	sort(events.begin(), events.end());

	int cntwithseat = 0;
```

```
set< pair<int, int> > posofset;



for(auto [time, t]: events) {
        if(t < 0) {
                // remove
                if(posof[-t].first >= 0) {
                        posofset.erase(posof[-t]);
                        posof[-t] = make_pair(-1, -1);
                        cntwithseat--;
                }
        } else {
                // add
                if(cntwithseat == 0) {
                        posof[t] = make_pair(0, 0);
```

```
            posofset.insert(posof[t]);

            cntwithseat++;

            cout << "1 1\n";

            continue;

    }



    vector< pair<int, int> > uux0(posofset.begin(), posofset.end());



    int maxdist = 0;

    int besty = -2;

    int bestx = -2;

    for(int y = 0; y < n; y++) {

            int vvptr = 0;

            for(auto [vx, y0]: uux0) {

                    int vy = abs(y - y0);
```

```
                if(vvptr > 0 && vy >= vv[vvptr - 1].vy +
abs(vv[vvptr - 1].vx - vx)) {

                        continue;

                }

                while(vvptr > 0 && vv[vvptr - 1].vy >= vy +
abs(vx - vv[vvptr - 1].vx)) {

                        vvptr--;

                }

                vv[vvptr] = Tick{vx, vy, 0};

                if(vvptr > 0) {

                        auto vx1 = vv[vvptr - 1].vx;

                        auto vy1 = vv[vvptr - 1].vy;

                        int l = (vy - vy1 + vx + vx1 + 1) / 2;

                        vv[vvptr].l = l;

                }

                vvptr++;

        }
```

```
vv[vvptr].l = m;

for(int i = 0; i < vvptr; i++) {

        int x = vv[i].l;

        int dist = vv[i].vy + abs(vv[i].vx - x);

        if(dist > maxdist) {

                maxdist = dist;

                besty = y;

                bestx = x;

        }



        x = vv[i + 1].l - 1;

        dist = vv[i].vy + abs(vv[i].vx - x);

        if(dist > maxdist) {

                maxdist = dist;

                besty = y;
```

```
                                        bestx = x;
                                    }
                                }
                            }
                            if(besty >= 0) {
                                posof[t] = make_pair(bestx, besty);
                                posofset.insert(posof[t]);
                                cntwithseat++;
                            }
                            cout << besty + 1 << " " << bestx + 1 << "\n";
                        }
                    }


                    return 0;
                }
```

Задача 4.

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

using namespace std;

const int NORTHBOUND = 1;
const int SOUTHBOUND = 2;
const int EASTBOUND = 3;
const int WESTBOUND = 4;

bitset<10800 * 2> field[5400 * 2];

int lat, lon;
int orient = NORTHBOUND;
bool south_pole_visited = false;
bool north_pole_visited = false;
```

```
int mod(int a, int b) {
	return (a % b + b) % b;
}


void addpoint() {
	if(lat == -5400) {
		south_pole_visited = true;
	} else if(lat == 5400) {
		north_pole_visited = true;
	} else {
		for(int dlat: {-1, 0}) {
			for(int dlon: {-1, 0}) {
				field[mod(lat + dlat, 5400 * 2)][mod(lon + dlon, 10800 *
2)] = true;
			}
```

```
                }

        }

}


void handle(char c, int cnt) {

        if(c == 'F') {

                if(orient == NORTHBOUND) {

                        for(int i = 0; i < cnt; i++) {

                                if(lat == 5400) { // north pole

                                        orient = SOUTHBOUND;

                                        lat--;

                                } else {

                                        lat++;

                                }

                                addpoint();
```

```
            }
        } else if(orient == SOUTHBOUND) {
            for(int i = 0; i < cnt; i++) {
                if(lat == -5400) { // south pole
                    orient = NORTHBOUND;
                    lat++;
                } else {
                    lat--;
                }
                addpoint();
            }
        } else if(orient == EASTBOUND) {
            for(int i = 0; i < cnt; i++) {
                if(lon == 10800) {
                    lon = -10800;
```

```
                }

                lon++;

                addpoint();

        }

} else if(orient == WESTBOUND) {

        for(int i = 0; i < cnt; i++) {

                lon--;

                if(lon == -10800) {

                        lon = 10800;

                }

                addpoint();

        }

} else {

        assert(false);

}
```

```
} else if(c == 'L') {
        for(int i = 0; i < cnt % 4; i++) {
                if(orient == NORTHBOUND) {
                        orient = WESTBOUND;
                } else if(orient == WESTBOUND) {
                        orient = SOUTHBOUND;
                } else if(orient == SOUTHBOUND) {
                        orient = EASTBOUND;
                } else if(orient == EASTBOUND) {
                        orient = NORTHBOUND;
                } else {
                        assert(false);
                }
        }
} else if(c == 'R') {
```

```
            for(int i = 0; i < cnt % 4; i++) {
                if(orient == NORTHBOUND) {
                    orient = EASTBOUND;
                } else if(orient == EASTBOUND) {
                    orient = SOUTHBOUND;
                } else if(orient == SOUTHBOUND) {
                    orient = WESTBOUND;
                } else if(orient == WESTBOUND) {
                    orient = NORTHBOUND;
                } else {
                    assert(false);
                }
            }
        } else {
            assert(c == 'S');
```

```
        }
}


int main() {
        double radius;
        cin >> lat >> lon >> radius;


        addpoint();


        string s;
        getline(cin, s);
        getline(cin, s);


        int cnt = 0;
        for(char c: s) {
```

```
            if(c == ' ') {
                continue;
            } else if('0' <= c && c <= '9') {
                cnt = cnt * 10 + (c - '0');
            } else {
                if(cnt == 0) {
                    cnt = 1;
                }
                handle(c, cnt);
                cnt = 0;
            }
        }

        int ans = north_pole_visited + south_pole_visited;
        for(auto& row: field) {
```

```
            ans += row.count();
        }

        double area = 2 * M_PI * M_PI * ans * radius * radius / (5400 * 2 * 10800 * 2);
        cout << fixed << setprecision(10) << ans << " " << area << endl;
        return 0;
}
```

Задача 5.

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

using namespace std;

const long long POW = 75928365;
const long long MOD = 1000000000000000013LL;
```

```
long long hashvec(const vector<int>& a) {
	long long h = 0;
	for(int x: a) {
		h = ((__int128)h * POW + x) % MOD;
	}
	return h;
}



int main() {
	int n;
	cin >> n;


	vector<int> a(n + 1);
	for(int i = 0; i < n + 1; i++) {
```

```
        cin >> a[i];
}

vector<int> prefix;
vector<long long> prefixeshashes(n + 1, 0);
vector<long long> prefixeshashes1(n + 1, 0);

vector<int> last = a;
sort(last.begin(), last.end());
for(int j = 0; j < n + 1; j++) {
        prefixeshashes[j] = last[j] % MOD;
}
prefix.push_back(last[0]);

long long curpow = 1;
```

```
        for(int i = 1; i < n; i++) {

                vector< pair<long long, int> > cont;

                for(int j = 0; j < n + 1; j++) {

                        long long hash = ((__int128)a[j] * curpow + prefixeshashes1[j]) %
MOD;

                        cont.emplace_back(hash, last[j]);

                }

                sort(cont.begin(), cont.end());



                vector<int> exp;

                for(int j = 0; j < n + 1; j++) {

                        exp.push_back(j);

                }

                sort(exp.begin(), exp.end(), [&](int j1, int j2) {

                        if(prefixeshashes[j1] == prefixeshashes[j2]) {

                                return j1 < j2;
```

```
                } else {
                        return prefixeshashes[j1] < prefixeshashes[j2];
                }
        });

        for(int jj = 0; jj < n + 1; jj++) {
                int j = exp[jj];
                last[j] = cont[jj].second;
                prefixeshashes1[j] = prefixeshashes[j];
                prefixeshashes[j] = ((__int128)prefixeshashes[j] * POW + last[j])
% MOD;
        }
        prefix.push_back(last[0]);
        curpow = (__int128)curpow * POW % MOD;
    }
```

```
    prefix.push_back(a[0]);

    int zerooffset = find(prefix.begin(), prefix.end(), 0) - prefix.begin();

    vector<int> vec(prefix.begin() + zerooffset + 1, prefix.end());

    vec.insert(vec.end(), prefix.begin(), prefix.begin() + zerooffset);



    for(auto x: vec) {

        cout << x << " ";

    }

    cout << endl;



    return 0;

}
```

Задача 6.

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



using namespace std;
```

```
int main() {
	int n, k;
	cin >> n >> k;


	vector<int> t(n);
	int maxti = 0;
	vector<int> cntgoingonline(1000001);
	for(int i = 0; i < n; i++) {
		cin >> t[i];
		if(i > 0) {
			maxti = max(maxti, t[i]);
			cntgoingonline[t[i]]++;
		}
	}
```

```
int network_data = 0;

int curtm = 0;

int cntonline = 0;

while(network_data < k) {

        cntonline += cntgoingonline[curtm];

        if(cntonline == n - 1 && network_data + 1 >= k) {

                cout << curtm + 1 << endl;

                return 0;

        }

        network_data = min(network_data + cntonline, k);

        curtm++;

}



cout << max(curtm, maxti) + 1 << endl;
```

```
        return 0;

}
```