ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

Факультет программной инженерии и компьютерной техники

Кафедра «ВТ»

ЛАБОРАТОРНАЯ РАБОТА № _3_

ПО ДИСЦИПЛИНЕ «*Алгоритмы и структуры данных*»

Выполнил(а): Чан Чунг Дык

Группа: Р3202

Санкт-Петербург

2019

1067. Disk Tree

Algorithm :

+ We just need to create a tree.
+ Suppose a folder is a tree, so it contains value as string and array of subfolder as tree.

```
#include <iostream>
#include <string>
#include <cstddef>
#include <map>

using namespace std;
const int maxn = 100;
int n;

struct tree {
  tree() {
    this->value = "";
    this->child = {};
  }

  tree(string value) {
    this->value = value;
    this->child = {};
  }

  tree(string value, map<string, tree *> child) {
    this->value = value;
    this->child = child;
  }
  string value;
  map<string, tree *> child;
};

void output(tree *root, int tab) {
  for (int i = 0; i < tab; i++)
    cout << " ";
  if (root->value != "") {
    cout << root->value << endl;
  }
  tab++;
  for (auto &i : root->child) {
    output(i.second, tab);
  }
}

int main() {
  cin >> n;
  tree *root = new tree();

  for (int i = 0; i < n; i++) {
    string s;
    string value = "";
    cin >> s;
```

```
    tree *tmp = root;
    for (int j = 0; j <= s.length(); j++) {

      if (s[j] == '\\' || s[j] == '\0') {
        auto dir_value = tmp->child.find(value);
        if (dir_value == tmp->child.end()) {
          tree *new_node = new tree(value);
          tmp->child.insert(pair<string, tree *>(value, new_node));
          tmp = tmp->child.find(value)->second;
        } else {
          tmp = dir_value->second;
        }
        value = "";
      } else
        value += s[j];
    }
  }

  output(root, -1);
}
```

1494. Monobilliards

Algorithm : in each time, the inspector takes out the ball, it will be maximum ball in the table's pocket. So we use stack to save the balls, which are in pocket. When the inspector takes out the ball called x. if x < the first ball in stack, it means mr. Chichikov is a cheater, if not, we check again with other balls.

```
#include <stack>
#include <iostream>
#include <vector>

using namespace std;

const int64_t maxn = 100000 + 10;
int64_t n;

vector<int64_t> arr;
stack<int64_t> st;

void input() {
  cin >> n;
  for (int64_t i = 0; i < n; i++) {
    int64_t x;
    cin >> x;
    arr.push_back(x);
  }
```

```
}
void solve() {
  int64_t i = 1;
  int64_t j = 0;
  st.push(i);
  while (j < n) {
    if (!st.empty() && (arr[j] == st.top())) {
      j++;
      st.pop();
    } else if (!st.empty() && (arr[j] < st.top())) {
      cout << "Cheater";
      return;
    } else {
      i++;
      st.push(i);
    }
  }
  cout << "Not a proof";
}

int main() {
  input();
  solve();
}
```

1521. War games 2

Suppose the current solider is x. the next solider, who leave the circle will be x+k. We will check in range [x,x+k], how many soliders was left. If there are y soliders left, we will check to solider (x+k+y). we check until k soliders, who didn't leave after solider x.

Using BIT tree to optimize time.

```
#include <iostream>

using namespace std;

const int maxn = 100000 + 10;
int n, k_first;
int m;
int arr[2 * maxn];
int F[2 * maxn];
void input() {
  cin >> n >> k_first;
  m = 2 * n;
  for (int i = 1; i <= m; i++)
```

```
    arr[i] = 1;
  for (int i = 0; i <= m; i++)
    F[i] = 0;
}

void update_tree(int i, int val) {
  while (i <= m) {
    F[i] = F[i] + val;
    i = i + (i & -i);
  }
}
// tinh sl soldat tu khoang i->j
int calc_sum(int i, int j) {
  int result = 0;
  while (j >= i) {
    if (j - (j & -j) >= i) {
      result += F[j];
      j = j - (j & -j);
    } else {
      result += arr[j];
      j = j - 1;
    }
  }
  return result;
}

// tinh sum 1->i
int calc_sum_left(int i) {
  int result = 0;
  while (i >= 1) {
    result = result + F[i];
    i = i - (i & -i);
  }
  return result;
}
// find next i
int post(int x) {
  if (x > n)
    x -= n;
  while (arr[x] != 1) {
    x++;
    if (x > n)
      x = 1;
  }
  return x;
}
```

```
void solve() {
  for (int i = 1; i <= m; i++)
    update_tree(i, arr[i]);
  int count = n;
  int i = 1;
  int k;
  while (count > 1) {
    k = k_first;
    if (count < k) {
      k = k % count;
      if (k == 0)
        k = count;
    }
    int j = i + k - 1;
    int tmp = calc_sum(i, j);
    while (tmp != k) {
      j += k - tmp;
      tmp = calc_sum(i, j);
    }

    j = j % n;
    if (j == 0)
      j = n;

    cout << j << " ";
    arr[j] = 0;
    update_tree(j, -1);
    arr[j + n] = 0;
    update_tree(j + n, -1);
    i = post(j + 1);

    //----------------
    count--;
  }
  for (int i = 1; i <= n; i++)
    if (arr[i] == 1)
      cout << i;
}

int main() {
  input();
  solve();
}
```