Национальный исследовательский университет ИТМО
(Университет ИТМО)

Факультет систем управления и робототехники

Дисциплина: Алгоритмы и структуры данных
Отчет по практической работе (1521 задача).

Студент:
Евстигнеев Дмитрий
Группа: *R3242*
Преподаватель:
Тропченко Андрей Александрович

Санкт-Петербург
2021

Цель: написать программу для решения задачи № на сайте Timus Online

Задача:

1521. Военные учения 2

Ограничение времени: 1.0 секунды

Ограничение памяти: 64 МБ

Вступление

В ходе недавних военных учений (более подробно эта история рассказана в задаче «Военные учения») министр обороны Советской Федерации товарищ Иванов имел возможность лично убедиться в блестящей боевой готовности солдат вверенной ему Советской Армии. Но одна вещь всё же продолжала беспокоить выдающегося военачальника. Прославленный генерал понимал, что была продемонстрирована лишь физическая подготовка солдат. Теперь настало время организовать очередные учения и проверить интеллектуальные способности личного состава.

Генерал Шульман, вновь назначенный ответственным за проведение учений, пожертвовал все выделенные деньги бедным и с чистой совестью лёг спать. Во сне генералу явился учебник по тактике и изложил схему, руководствуясь которой можно провести учения совершенно бесплатно.

Задача

В соответствии с этой схемой учения делятся на **N** раундов, в течение которых **N** солдат, последовательно пронумерованных от 1 до **N**, маршируют друг за другом по кругу, т.е. первый следует за вторым, второй за третьим, ..., (**N**-1)-й за **N**-м, а **N**-й за первым. В каждом раунде очередной солдат выбывает из круга и идёт чистить унитазы, а оставшиеся продолжают маршировать. В очередном раунде выбывает солдат, марширующий на **K** позиций впереди выбывшего на предыдущем раунде. В первом раунде выбывает солдат с номером **K**.

Разумеется, г-н Шульман не питал никаких надежд на то, что солдаты в состоянии сами определить очерёдность выбывания из круга. «Эти неучи даже траву не могут ровно покрасить», – фыркнул он и отправился за помощью к прапорщику Шкурко.

Исходные данные

Единственная строка содержит целые числа **N** ($1 \le N \le 100000$) и **K** ($1 \le K \le N$).

Результат

Вывести через пробел номера солдат в порядке их выбывания из круга.

Пример

исходные данные	результат
5 3	3 1 5 2 4

Принято системой (JUDGE_ID: 231802FR):

9376852	12:33:58 24 май 2021	Dmitry Evstigneev	1521. Военные учения 2	G++ 9.2 x64	Accepted

Решение на языке C++:

```
#include
<vector>
#include <algorithm>

class List
{
    std::vector<int> v;
    int size, bottom, step, current, remains;


    inline int nextpow(int x)
    {
        x--;
        for(int i = 1; i <= 16; i *= 2)
            x |= x >> i;
        return x + 1;
    }

    int log(int x)
    {
        int ret = 0;
        while(x /= 2)
            ret++;
        return ret;
    }

    inline int pow2(int n)
    {
        int ret = 1;
        while(n-- > 0)
            ret *= 2;
        return ret;
    }

    inline int parent(const int& node) const
    {
        return node/2;
    }

    inline int rightchild(const int& node) const
    {
        return node*2 + 1;
    }

    inline int leftchild(const int& node) const
    {
```

```
        return node*2;
    }

    inline bool intree(const int& node) const
    {
        return node <= bottom + size && node > 0;
    }

public:
    List(int n, int step) : size(n), step(step), current(step), remains(n)
    {
        v.resize(nextpow(n*2));
        int height = log(n*2 - 1);
        bottom = pow2(height) - 1; // The node immediately preceding the first
leaf

        for(int i = bottom + 1; i <= bottom + n; i++) // Leafs have key 1
            v[i] = 1;
        for(int i = bottom; i >= 1; i--) // Internal nodes' keys are the sum
their childrens keys
            v[i] = (intree(leftchild(i)) ? v[leftchild(i)] : 0)
                + (intree(rightchild(i)) ? v[rightchild(i)] : 0);
    }

    int remaining()
    {
        return remains;
    }

    int getNext()
    {
        if(remains-- == size) // We're already at the first one so we handle
this one directly
            return current;

        int node = bottom + current;
        while(node) // Bubble up a subtraction through the tree
        {
            v[node]--;
            node = parent(node);
        }
        node = bottom + current;

        int k = step; // The remaining amount of soldiers that we need to skip
        enum source { left, right, up }; // The direction we arrived from to a
node
        source source = right;
```

```
        // Our main routine - start at a leaf, search our way through the tree
until we have
        // skipped the amount of soldiers that need to be skipped
        while(k)
        {
            int lc = leftchild(node);
            int rc = rightchild(node);
            if(source == up)
            {
                if(intree(lc) && k > v[lc]) // We can skip the entire left
subtree
                {
                    k -= v[lc];
                    source = up;
                    node = rc;
                }
                else if(!intree(lc) && k == v[node]) // Final leaf found
                    k--;
                else // The answer lies in our left subtree, enter it
                {
                    source = up;
                    node = lc;
                }
            }
            else if(source == right) // Always go up when we've come from a
right subtree
            {
                source = node == rightchild(parent(node)) ? right : left;
                node = parent(node);
            }
            else // Coming from the left
            {
                if(intree(rc) && k > v[rc]) // We can skip the entire subtree in
this case
                {
                    source = node == rightchild(parent(node)) ? right : left;
                    k -= v[rc];
                    node = parent(node);
                }
                else // Our answer lies within the right subtree; we must pursue
our destiny there
                {
                    node = rc;
                    source = up;
                }
            }
            if(!intree(node))    // This happens whenever we started somewhere
and couldn't find k
```

```
            {                   // successors to the right of us in the tree -
in this case, we
                node = bottom+1; // wrap around from the leftmost soldier leaf
                k -= v[node];
                source = right;
            }
        }
        return current = node-bottom; // Transform from tree index to soldier
number
    }
};

int main()
{
    int N, K;
    scanf("%d %d", &N, &K);

    List list(N, K);
    while(list.remaining())
    {
        int ans = list.getNext();
        // So this is just a faster way of printing a number than having to call
printf and
        // do their generalized stuff. It shaves off another 0.1s from the
solution
        char s[9];
        int l = 0;
        while (ans)
        {
            s[l++] = ans % 10;
            ans /= 10;
        }
        while (l--)
            putc('0' + s[l], stdout);
        putc(' ', stdout);
    }
    return 0;
}
```

Суть алгоритма:

Создайте полное дерево с n листами, имеющими ключи 1 (соответствующие нашим солдатам), и каждому другому узлу, имеющему в качестве ключа сумму двух его потомков. Каждый раз, когда мы вычисляем «преемника» (как определено в задаче) какого-либо солдата, мы начинаем с его листа, всплываем на вычитание по дереву и начинаем обход дерева от листа с

помощью узловых ключей, чтобы помочь нам найти его преемник - в каждом узле дерева мы можем использовать его ключ, чтобы пропустить некоторое количество солдат, не посещая левое поддерево. Например, если нам нужно пропустить дополнительных 5 солдат, и мы находимся в узле, левый дочерний элемент которого имеет ключ 3, мы пропускаем это поддерево и продолжаем поиск в правом поддереве, чтобы остальные 2 солдата пропустили его. Решение несколько длинное, есть и более короткие.