Kiểm tra tính đúng đắn và hiệu năng của chương trình bằng bộ test

Nội dung

01

Bộ test là gì? Tại sao cần bộ test?

Bộ test

Tập hợp nhiều cặp input-output của một bài toán nào đó

Trong đó input và output phải thỏa mãn yêu cầu bài toán

Tại sao cần bộ test

Dùng để kiểm tra tính đúng đắn và hiệu năng của chương trình

Đảm bảo chương trình chạy theo ý muốn của người lập trình

Testing bao gồm

Test script:

Là chương trình test tự động

Test Cases:

Là các mẫu input output nhất định

Test Scenarios:

Là 1 chuỗi testcase, dùng đề biểu diễn 1 tình huống

Test Steps:

Để mô tả các bước thực hiện và kết quả mong đợi ở mỗi bước

Các thành phần của Test-case

1. Corner cases

Corner cases nghĩa là những trường đặt biệt của đề có thể làm lời giải chạy sai

Để kiểm tra lời giải có thể giải quyết mọi trường hợp

Thường được tạo bằng tay

Các thành phần của Test-case

2. Boundary cases

Boundary cases bao gôm những trường hợp giới hạn của đề, những rằng buột đặc biệt

Để kiểm tra lời giải đã đảm bảo hết những rằng buộc của đề hay chưa

Thường được tạo bằng chương trình khác chương trình sinh test

Các thành phần của Test-case

3. Huge Test-case

Huge Test-case bao gôm những bộ test lớn, trường hợp xấu nhất

Để kiểm tra thời gian thực thi của lời giải đã đảm bảo trong giới hạn đề hay chưa

Thường được tạo bằng chương trình khác chương trình sinh test

02

Tính đúng đắn và hiệu năng

Tính đúng đắn

Chương trình chạy ra output cần tìm => Thuật toán đúng

```
Test 1
0.00 s and 0 KiB
ACCEPT
Test 2
0.00 s and 0 KiB
ACCEPT
Test 3
0.00 s and 0 KiB
ACCEPT
Test 4
0.00 s and 0 KiB
```

```
Test 1
0.00 s and 0 KiB
ACCEPT
Test 2
0.00 s and 0 KiB
WRONG
Test 3
0.00 s and 0 KiB
WRONG
Test 4
0.00 s and 0 KiB
```

Hiệu năng

Mức độ tiêu tốn thời gian và không gian dữ liệu của thuật toán

Time Limit:	1 secs
Source Limit:	50000 Bytes

Ví dụ:

- Giả sử dùng thuật toán QuickSort để sắp xếp 1000 phần tử, tốn ít nhất 200ms.

Quick Sort: O(nlogn)

- Thuật toán Interchange Sort sẽ mất 66,7s để hoàn thành.

=> Thuật toán Quick Sort trong trường hợp này có hiệu năng cao hơn thuật toán Interchange Sort

Câu hỏi:
Giả sử có một trường hợp cả hai thuật toán Quick Sort và Merge Sort đều có thời gian thực thi như nhau. Vậy thuật toán nào có hiệu năng cao hơn?

Công cụ test

Một số trang web như Hackerrank, Leetcode, Wecode cho phép tạo bài toán và thêm bộ test. Ta có thể tận dụng để test cho chương trình mà không cần tạo trình chấm.

Trong thực tế, để kiểm thử cho ứng dụng web, phần mềm, cơ sở dữ liệu, API. Người ta sử dụng các phần mềm chuyên dụng như Selenium, TestingWhiz, TestComplete, …

Kiểm thử phần mềm trong thực tế

- Là quá trình thực thi 1 chương trình với mục đích tìm ra lỗi.
- Đảm bảo sản phẩm phần mềm đáp ứng chính xác, đầy đủ và đúng theo yêu cầu của khách hàng, yêu cầu của sản phẩm đề đã đặt ra.
- Cung cấp mục tiêu, cái nhìn độc lập về phần mềm, điều này cho phép việc đánh giá và hiểu rõ các rủi ro khi thực thi phần mềm.
- Tạo điều kiện cho bạn tận dụng tối đa tư duy đánh giá và sáng tạo để bạn có thể phát hiện ra những điểm mà người khác chưa nhìn thấy.

Quy trình test/kiểm thử trong thực tế

Quy trình	Đầu vào	Đầu ra
Phân tích yêu cầu	Tài liệu SRS, tài liệu thiết kế, bản prototype	File Q & A
Lập kế hoạch	Các tài liệu đã được cập nhật thông qua file Q & A trong giai đoạn phân tích yêu cầu	Test plan, checklist
Thiết kế kiểm thử	Test plan, checklist và các tài liệu đặc tả đã được cập nhật	Test design, test case, check list, test data, test automation script

Quy trình	Đầu vào	Đầu ra
Chuẩn bị môi trường	Test plan, smoke test case, test data	Môi trường đã được chuẩn bị sẵn sàng cho việc test và các kết quả của smoke test case
Thực hiện kiểm thử	Test design, test case, check list, test data, test automation script	Test results, defect reports
Kết thúc	Tất cả các tài liệu được tổng hợp từ giai đoạn đầu tiên	Test report, test results final

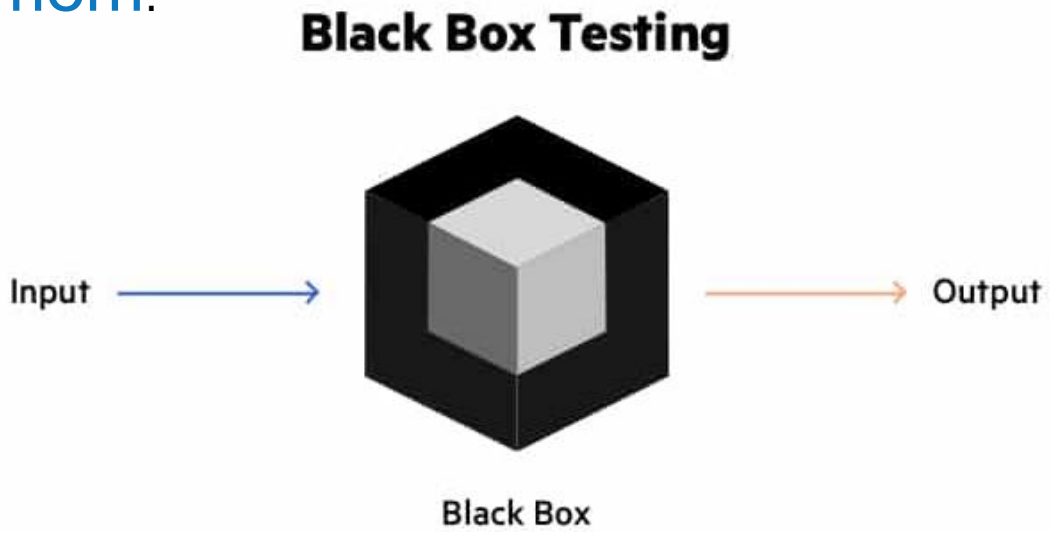
Một số khái niệm về kiểm thử

Một số khái niệm về kiểm thử

- Kiểm thử hộp đen
- Kiểm thử hộp trắng
- Kiểm thử tích hợp
- Kiểm thử hệ thống
- Kiểm thử chức năng
- Kiểm thử phi chức năng
- .....

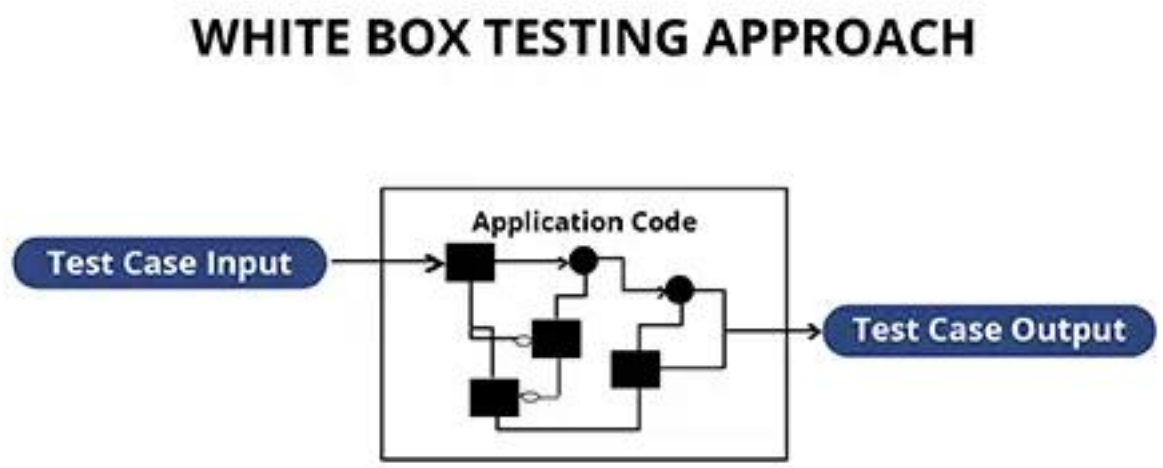
Kiểm thử hộp đen (Black box testing)

- Là phương pháp kiểm thử mà tester sẽ chỉ xem xét đến input và output của chương trình mà không quan tâm code bên trong được viết ra sao.
- Tester thực hiện kiểm thử dựa hoàn toàn vào đặc tả yêu cầu.
- Mục đích của kiểm thử hộp đen là tìm ra các lỗi ở giao diện, chức năng của phần mềm.



Kiểm thử hộp trắng (White box testing)

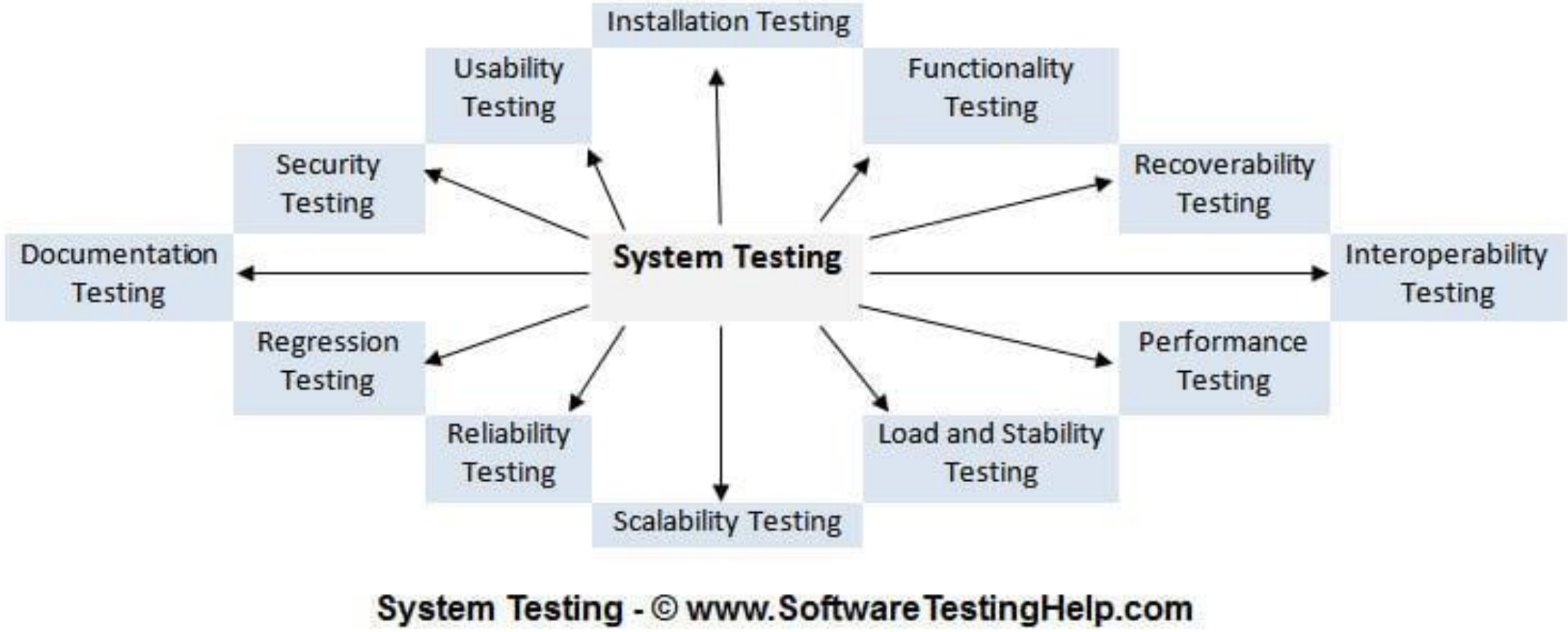
Là phương pháp kiểm thử mà cấu trúc thuật toán của chương trình được đưa vào xem xét. Các trường hợp kiểm thử được thiết kế dựa vào cấu trúc mã hoặc cách làm việc của chương trình. Người kiểm thử truy cập vào mã nguồn của chương trình để kiểm tra nó.



Kiểm thử hộp trắng (White box testing)

Kiểm thử 1 hệ thống đã được tích hợp hoàn chỉnh để xác minh rằng nó đáp ứng được yêu cầu Kiểm thử hệ thống thuộc loại kiểm thử hộp đen . Kiểm thử hệ thống tập trung nhiều hơn vào các chức năng của hệ thống . Kiểm tra cả chức năng và giao diện , các hành vi của hệ thống 1 cách hoàn chỉnh, đáp ứng với yêu cầu.

Kiểm thử hộp trắng (White box testing)



03

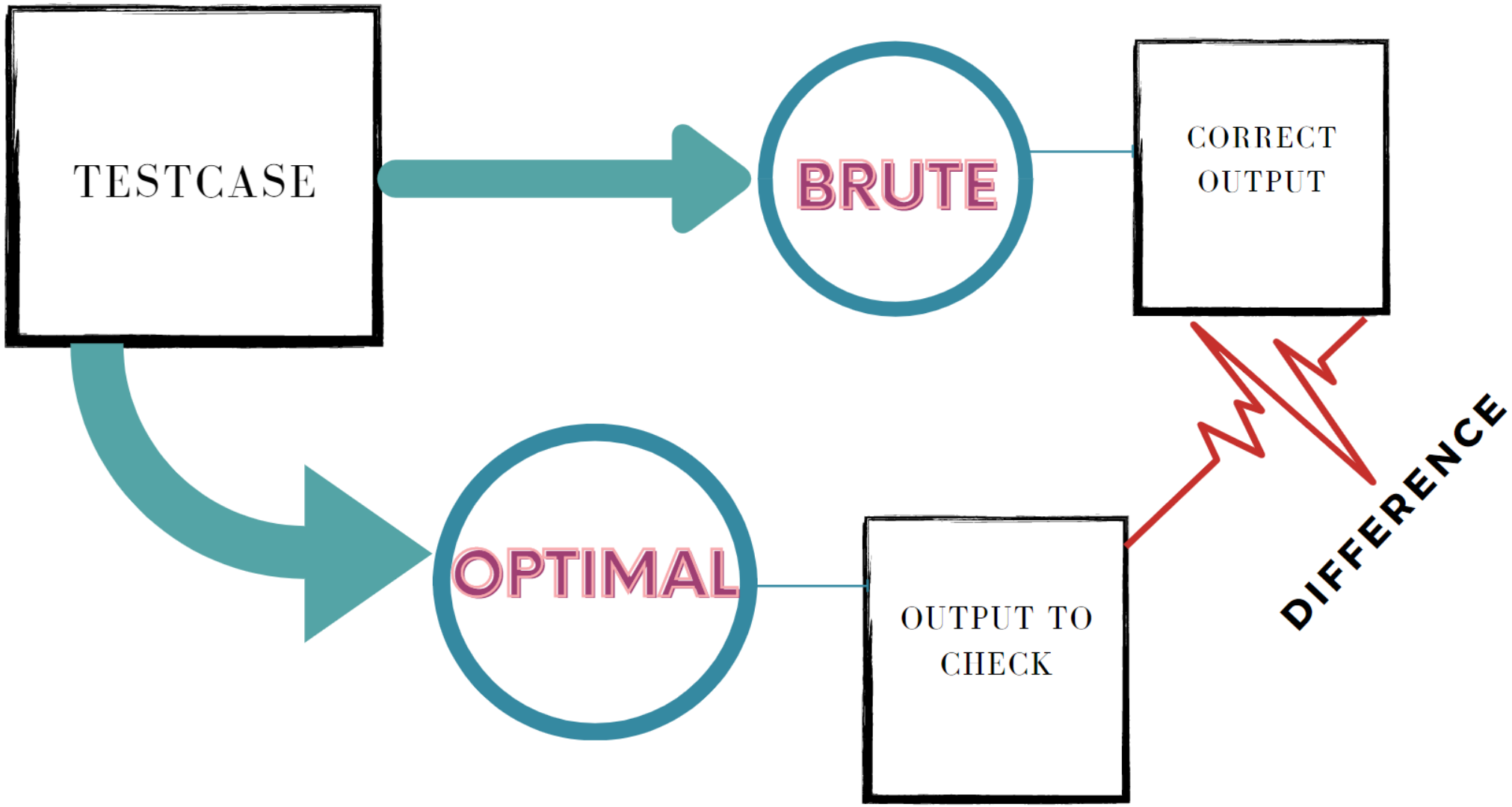
Cách test chương trình bằng trình chấm

Giới thiệu về trình chấm

Trình chấm là một bộ chương trình giúp bạn tự động sinh ra các test input ngẫu nhiên và tự động chạy hai chương trình lời giải khác nhau với các input đó để so sánh output.

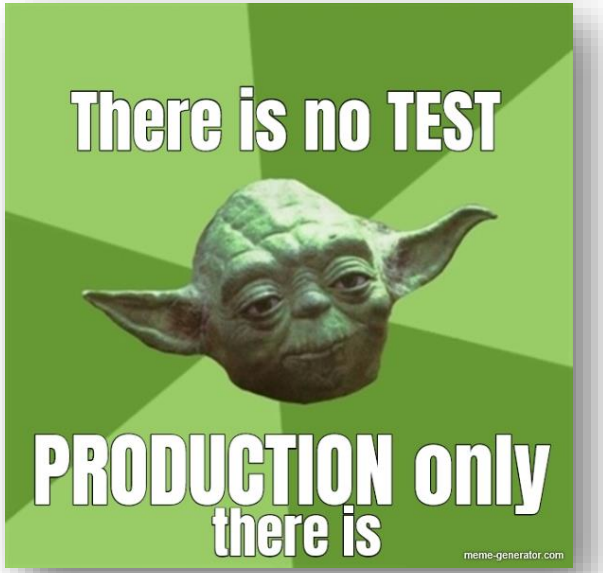
Trình chấm bao gồm 4 chương trình

- Trình sinh test
- Lời giải 1
- Lời giải 2
- Trình so test


TESTCASE
BRUTE
CORRECT OUTPUT
OPTIMAL
OUTPUT TO CHECK
DIFFERENCE

Trình sinh test

Trình sinh test là 1 chương trình sinh test ngẫu nhiên theo yêu cầu và giới hạn của đề



Một số hàm hữu ích để viết trình sinh test

Hàm sinh số ngẫu nhiên

randomNumber.cpp > ...

```cpp
#include<iostream>
#include <cstdlib>
#include <ctime>

using namespace std;

int randomNumber(int a, int b){
    return rand() % (b - a + 1) + a;
}

int main() {
    srand(time(NULL));
    cout << randomNumber(1, 100);
    return 0;
}
```

Hàm sinh mảng ngẫu nhiên

```cpp
int randomNumber(int a, int b){
    return rand() % (b - a + 1) + a;
}

void randomArray(int n, int *arr, int a, int b){
    for (int i = 0; i < n; i++){
        arr[i] = randomNumber(a,b);
    }
}

int main() {
    srand(time(NULL));
    cout << randomNumber(1, 100);
    return 0;
}
```

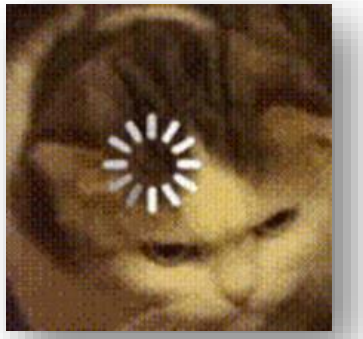
Một số hàm hữu ích để viết trình sinh test

Hàm sinh số nguyên Long Long ngẫu nhiên

```
109   mt19937 rd(chrono::steady_clock::now().time_since_epoch().count());
110   #define rand rd
111
112   long long randomLLNumber(long long l, long long h) {
113       assert(l <= h);
114       return l + rd() * 1LL * rd() % (h - l + 1);
115   }
```

Một số hàm hữu ích để viết trình sinh test

Vậy làm sao để sinh ra 1 mảng đã xắp sếp ngẫu nhiên ?



Một số hàm hữu ích để viết trình sinh test

Hàm sinh chuỗi ngẫu nhiên

randomString.cpp > ...

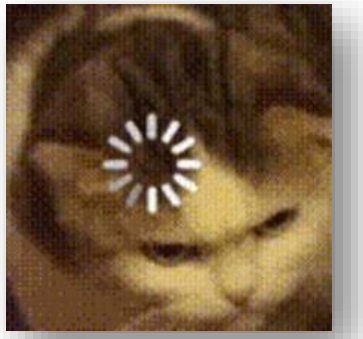
```cpp
#include<iostream>
#include <cstdlib>
#include <ctime>
#include <string>

using namespace std;

char randomCharacter(){
    return 'a' + rand() % 25;
}

string randomString(int len){
    string res = "";
    for (int i = 0; i < len; i++){
        res.push_back(randomCharacter());
    }
    return res;
}

```

```cpp
int main() {
    srand(time(NULL));
    cout << randomString(10);
    return 0;
}
```

```
PS D:\CODE STUFF\C++\CS112> ./randomString
waeddwkkyu
PS D:\CODE STUFF\C++\CS112> ./randomString
kivevbtnxv
PS D:\CODE STUFF\C++\CS112> ./randomString
kkkuwukkk
```

Một số hàm hữu ích để viết trình sinh test

Vậy làm sao để sinh ra 1 chuỗi palindrome?



Một số hàm hữu ích để viết trình sinh test

Hàm sinh cây không trọng số

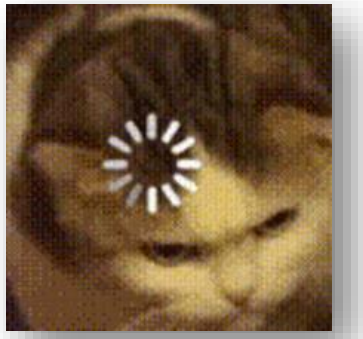
```cpp
set< pair<int,int> > unweighted_tree_generation(int n){
    vector< vector<int> > adj(n);
    set< pair<int,int> > container;
    for (int i = 0; i < n-1; i++){
        int u = rand() % n;
        int v =  rand() % n;
        pair<int,int> p = {u,v};
        adj[u].push_back(v);

        while (container.find(p) != container.end()
                    || isCyclic(n, adj) == true)
            {
                adj[u].pop_back();
                u = rand() % n;
                v = rand() % n;
                p = {u,v};
                adj[u].push_back(v);
            }
        container.insert(p);
    }
    return container;
}
```

```cpp
int main() {
    srand(time(NULL));
    int n = 6;
    set<pair<int,int>> container = unweighted_tree_generation(n);

    cout<< n << endl;
    set<pair<int, int>>::iterator it;
    for (it=container.begin(); it!=container.end(); ++it)
        cout<< it->first << " " << it->second << endl;
    return 0;
}
```

```
PS D:\CODE STUFF\C++\CS112> ./randomTree
6
0 3
1 5
2 3
3 4
4 5
```

Một số hàm hữu ích để viết trình sinh test

Vậy làm sao để sinh ra 1 cây có trọng số?



Một số hàm hữu ích để viết trình sinh test

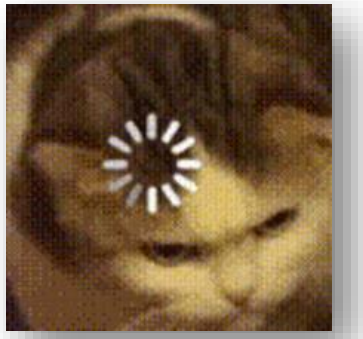
Hàm sinh đồ thị có hướng không trọng số

```
4   set< pair<int,int> >
5       directed_unweighted_graph_generation(int n, int maxEdge){
6       set< pair<int,int> > container;
7       for (int j=1; j<=maxEdge; j++)
8       {
9           int a = 1 + rand() % n;
10          int b = 1 + rand() % n;
11          pair<int, int> p = make_pair(a, b);
12
13          while (container.find(p) != container.end())
14          {
15              a = 1 + rand() % n;
16              b = 1 + rand() % n;
17              p = make_pair(a, b);
18          }
19          container.insert(p);
20      }
21      return container;
22  }
```

```
24  int main() {
25      srand(time(NULL));
26      int n = 6;
27      int m = 10;
28      set<pair<int,int>> container
29      = directed_unweighted_graph_generation(n, m);
30
31      cout<< n << " " << m <<  endl;
32      set<pair<int, int>>::iterator it;
33      for (it=container.begin(); it!=container.end(); ++it)
34          cout<< it->first << " " << it->second << endl;
35      return 0;
36  }
```

Một số hàm hữu ích để viết trình sinh test

Vậy làm sao để sinh ra 1 đồ thị vô hướng ?



Một số hàm hữu ích để viết trình sinh test

Vậy làm sao để sinh ra 1 đồ thị vô hướng ?

```
5   set< pair<int,int> >
6       undirected_unweighted_graph_generation(int n, int maxEdge){
7       set< pair<int,int> > container;
8       for (int j=1; j<=maxEdge; j++)
9       {
10          int a = 1 + rand() % n;
11          int b = 1 + rand() % n;
12          pair<int, int> p = make_pair(a, b);
13          pair<int, int> reverse_p = make_pair(b, a);
14          while (container.find(p) != container.end() ||
15                  container.find(reverse_p) != container.end() )
16          {
17              a = 1 + rand() % n;
18              b = 1 + rand() % n;
19              p = make_pair(a, b);
20              reverse_p = make_pair(b, a);
21          }
22          container.insert(p);
23      }
24      return container;
25  }
```

Lời giải 1 và lời giải 2

Lời giải 1 là lời giải tối ưu mà chúng ta cần kiểm tra về tính đúng đắn và hiệu suất

Lời giải 2 là lời giải đúng dễ dàng cài đặt không nhất thiết phải nhanh, thường sẽ cài bằng các thuật toán trâu

Tìm lời giải trâu cho bài toán sau

ĐƯỜNG BẬC THANG

Các bậc thang của đường dẫn lên đỉnh núi được đánh số từ 1 đến ***n***, 1 là bậc thang đầu tiên, ***n*** – bậc cuối cùng trên đỉnh núi. Ở các bậc có số chẵn 10 người ta ghi số của bậc trên bậc đó. Ngoài ra bậc đầu tiên và bậc cuối cùng cũng cũng được ghi số. Chi phí để sơn mỗi chữ số là như nhau và được coi là bằng 1.

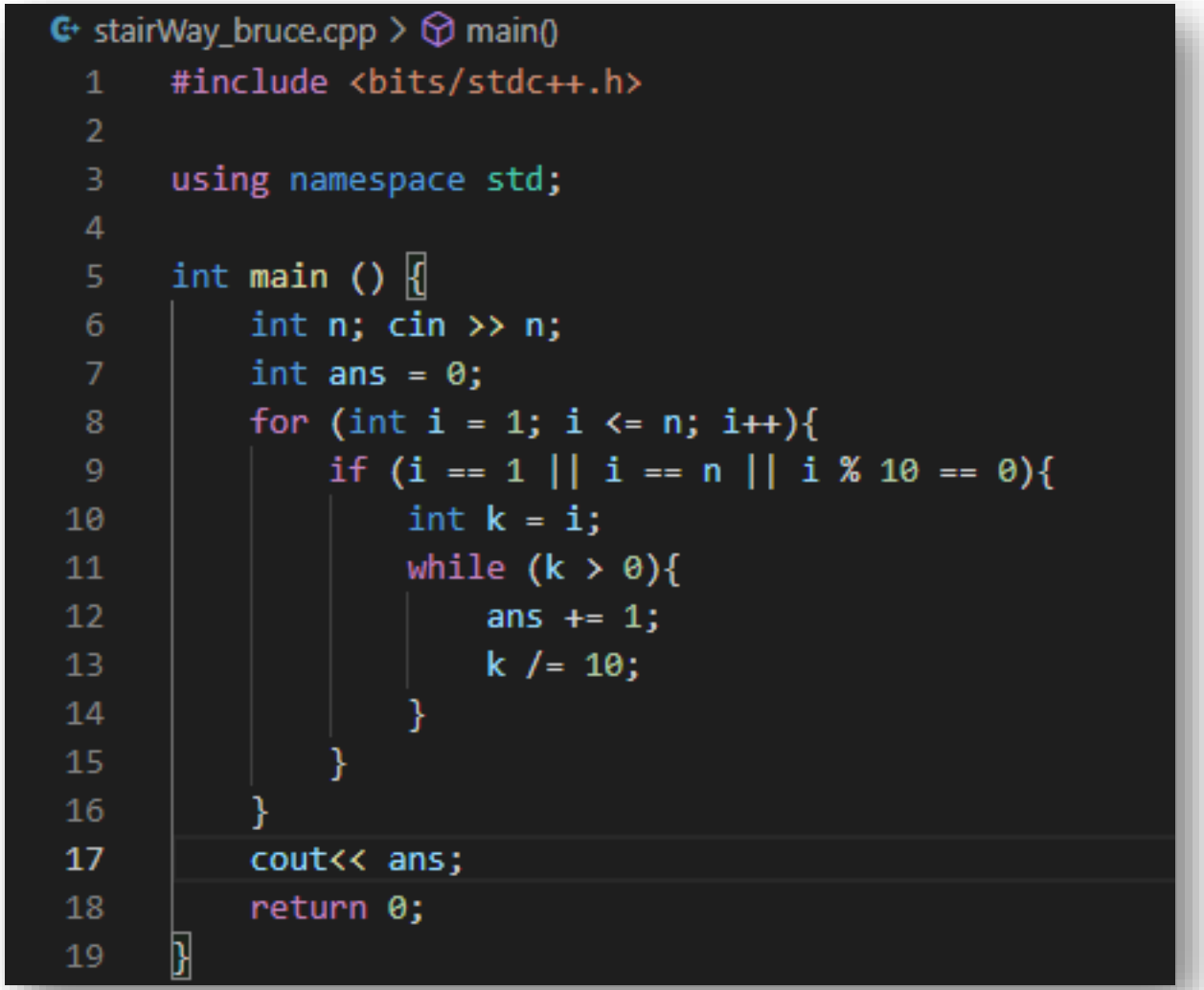
Hãy xác định tổng chi phí của việc đánh dấu trên toàn bộ đường đi.

Dữ liệu: Vào từ thiết bị nhập chuẩn gồm một dòng chứa số nguyên ***n*** ($1 \le n \le 10^{12}$).

Kết quả: Đưa ra thiết bị xuất chuẩn một số nguyên – tổng chi phí tìm được.

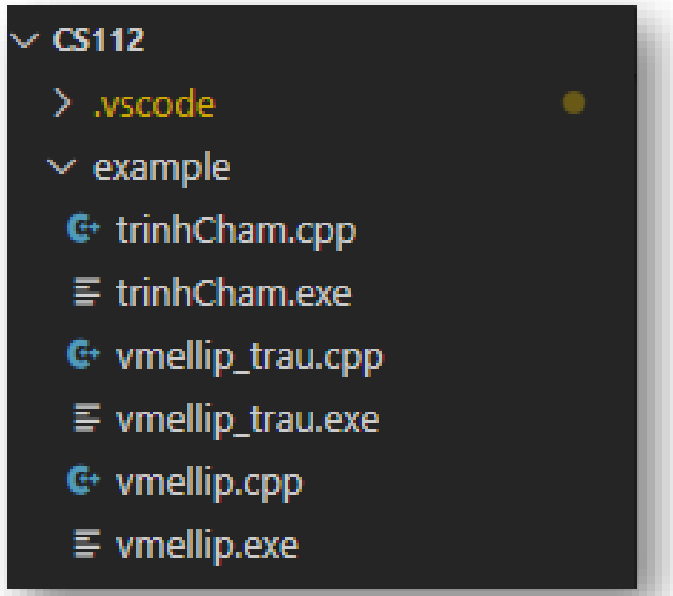
Ví dụ:

INPUT		OUTPUT
23		7

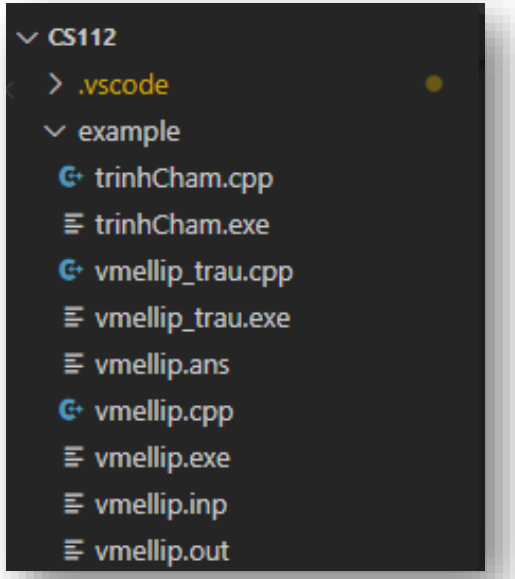

stairWay_bruce.cpp > main()
#include <bits/stdc++.h>
using namespace std;
int main () {
int n; cin >> n;
int ans = 0;
for (int i = 1; i <= n; i++){
if (i == 1 || i == n || i % 10 == 0){
int k = i;
while (k > 0){
ans += 1;
k /= 10;
}
}
}
cout<< ans;
return 0;
}

Trình so test

Chương trình này có nhiệm vụ duyệt N lần, mỗi lần duyệt thì chạy trình sinh test rồi chạy 2 chương trình lời giải 1 và lời giải 2 so sánh output mà đưa ra kết quả



Template cho trình chấm C++




```cpp
int main()
{
    string NAME = "baitap";
    int NTEST = 100;
    srand(time(NULL));
    for (int iTest = 1; iTest <= NTEST; iTest++)
    {
        ofstream inp((NAME + ".inp").c_str());
        /***
            Code phần sinh test ở đây
        ***/
        inp.close();

        auto start = high_resolution_clock::now();
        system((NAME + ".exe").c_str());
        auto stop = high_resolution_clock::now();
        auto duration1 = duration_cast<microseconds>(stop - start);

        start = high_resolution_clock::now();
        system((NAME + "_trau.exe").c_str());
        stop = high_resolution_clock::now();
        auto duration2 = duration_cast<microseconds>(stop - start);

        if (system(("fc " + NAME + ".out " + NAME + ".ans").c_str()) != 0)
        {
            cout << "Test " << iTest << ": WRONG!\n";
            return 0;
        }
        cout << "Test " << iTest << ": CORRECT!\n";
        cout << NAME + ".exe" + "excution time: " << duration1.count()<< " microseconds" << endl;
        cout << NAME + "_trau.exe" + "excution time: " << duration2.count()<< " microseconds" << endl;
    }
    return 0;
}
```

Chương trình sẽ tự động sinh test và kiểm tra

```
PS D:\CODE STUFF\C++\CS112\example> ./trinhCham
Comparing files vmellip.out and VMELLIP.ANS
FC: no differences encountered

Test 1: CORRECT!
vmellip.exe excution time: 76797 microseconds
vmellip_trau.exe excution time: 56845 microseconds
Comparing files vmellip.out and VMELLIP.ANS
FC: no differences encountered

Test 2: CORRECT!
vmellip.exe excution time: 54852 microseconds
vmellip_trau.exe excution time: 39893 microseconds
Comparing files vmellip.out and VMELLIP.ANS
FC: no differences encountered

Test 3: CORRECT!
vmellip.exe excution time: 53855 microseconds
vmellip_trau.exe excution time: 35904 microseconds
Comparing files vmellip.out and VMELLIP.ANS
FC: no differences encountered
```

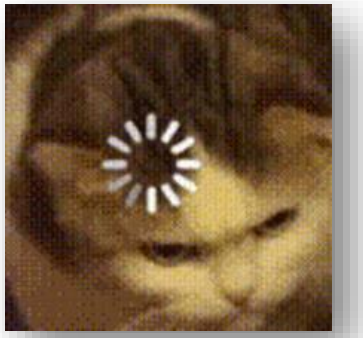
Chương trình sẽ dừng lại khi có 1 test không khớp

```
Test 16: CORRECT!
vmellip.exe excution time: 54852 microseconds
vmellip_trau.exe excution time: 35905 microseconds
Comparing files vmellip.out and VMELLIP.ANS
FC: no differences encountered

Test 17: CORRECT!
vmellip.exe excution time: 53856 microseconds
vmellip_trau.exe excution time: 35904 microseconds
Comparing files vmellip.out and VMELLIP.ANS
***** vmellip.out
81
24
***** VMELLIP.ANS
81
8
*****

Test 18: WRONG!
PS D:\CODE STUFF\C++\CS112\example> BAKA
```

04

Ví dụ

Ví dụ 1



Problem 7.08

Problem7.08

Nhập ma trận vuông A (n x n). Kiểm tra A có phải là ma trận Frobenius hay không.

Ma trận Frobenius được định nghĩa là ma trận vuông thoả các tính chất sau:

1. Tất cả các phần tử trên đường chéo chính bằng 1.
2. Có tối đa một cột mà các phần tử dưới đường chéo chính có thể nhận giá trị bất kì.
3. Ngoài ra, tất cả các phần tử khác có giá trị bằng 0.

INPUT

- 2 số nguyên là số dòng, số cột của ma trận A
- Giá trị các phần tử của ma trận A là số thực

OUTPUT

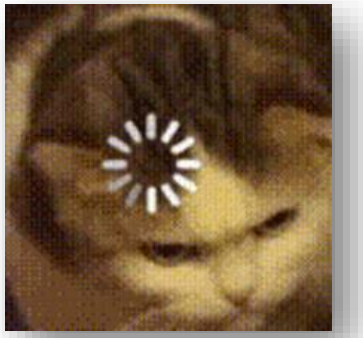
- "Yes", nếu A là ma trận Frobenius
- "No", nếu A không là ma trận Frobenius

Ví dụ 1

p7_8_test_generation.cpp > main()

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

using namespace std;

int main () {
    srand(time(NULL));
    int n = 1+ rand()%100;
    vector< vector<int> > A (n, vector<int> (n, 0));
    for (int i = 0; i < n; i++){
        A[i][i] = 1;
    }

    int k = rand() % (n-1);
    for (int i = k+1; i < n; i++){
        A[i][k] = rand()%100;
    }

    cout << n << " " << n << endl;
    for(int i = 0; i < n; i++){
        for (int j = 0; j < n; j++){
            cout<< A[i][j] << " " ;
        }
        cout<< endl;
    }
    return 0;
}
```

Ví dụ 2



Thầy Sơn mới thiết kế một hệ thống **N** máy tính (các máy tính được đánh số từ 1 đến **N**) được nối lại thành một mạng bởi **M** kênh nối, mỗi kênh nối hai máy nào đó và cho phép ta truyền tin từ máy này sang máy kia. Nhưng không may rằng Thầy Sơn đã bị 1 hacker nào đó tấn công vào một máy bất kì trong **N** máy. Để khắc phục sự cố thì thầy Sơn cần liệt kê ra các máy đã bị tấn công. Mặc dù vậy nhưng mà hệ thống máy tính của thầy rất lớn nên không thể liệt kê bằng cách thông thường được. Em hãy viết chương trình giúp thầy xác định các máy bị tấn công ? Biết rằng. nếu một máy bị cấn công thì các máy được kết nối trực tiếp cũng sẽ bị tấn công.

Dữ liệu:

- Dòng thứ nhất gồm số nguyên **N, M** ($1 \le N, M \le 10^6$). **N** là số máy tính và **M** là số kênh nối.
- **M** dòng tiếp theo, mỗi dòng gồm 2 số nguyên **u** và **v (1 <= u != v <= N) ,** cho biết máy **u** sẽ kết nối trực tiếp với máy **v.**
- Dòng cuối cùng chứa số nguyên **s (1 <= s <= N)**, là số hiệu của máy bị tấn công.

Kết quả:

- Dòng đầu tiên in ra số lượng các máy bị tấn công.
- Dòng thứ hai in ra dãy số hiệu các máy tính bị tấn công theo thứ tự tăng dần.

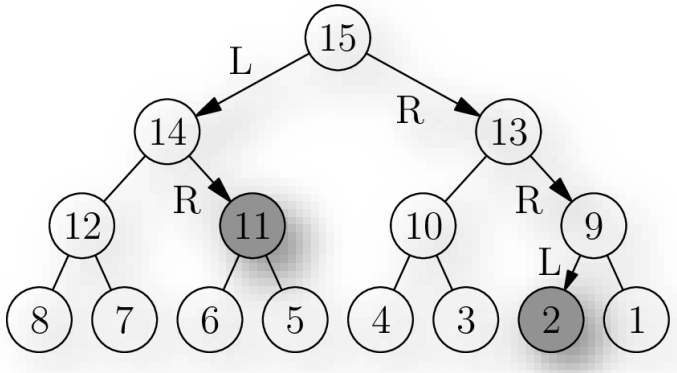
Ví dụ 2

ConnectedComponents_test_generation.cpp > main()

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

using namespace std;

set< pair<int,int> >
    undirected_unweighted_graph_generation(int n, int maxEdge){
    set< pair<int,int> > container;
    for (int j=1; j<=maxEdge; j++)
    {
        int a = 1 + rand() % n;
        int b = 1 + rand() % n;
        pair<int, int> p = make_pair(a, b);
        pair<int, int> reverse_p = make_pair(b, a);
        while (a == b && container.find(p) != container.end() ||
                container.find(reverse_p) != container.end() )
        {
            a = 1 + rand() % n;
            b = 1 + rand() % n;
            p = make_pair(a, b);
            reverse_p = make_pair(b, a);
        }
        container.insert(p);
    }
    return container;
}
```

```cpp
int main () {
    srand(time(NULL));
    int n = 1 + rand() % 1000000;
    int m = 1 + rand() % (min(100000, n*(n-1)/2));
    cout<< n << " " << m <<  endl;
    set<pair<int,int>>
        container = undirected_unweighted_graph_generation(n,m);

    set<pair<int, int>>::iterator it;
    for (it=container.begin(); it!=container.end(); ++it)
        cout<< it->first << " " << it->second << endl;
    return 0;
}
```

Ví dụ 3



Numbers On a Tree

Task

Lovisa's task is to calculate the label of a node, given the height of the tree H and the description of the path from the root.

Input

The only line of input contains the height of the tree H, $1 \leq H \leq 30$ and a string consisting of the letters 'L' and 'R', denoting a path in the tree starting in the root. The letter 'L' denotes choosing the left child, and the letter 'R' choosing the right child. The description of the path may be empty and is at most H letters.

Output

Output one line containing the label of the node given by the path.

Sample Input 1	Sample Output 1
3 LR	11
Sample Input 2	**Sample Output 2**
3 RRL	2
Sample Input 3	**Sample Output 3**
2	7

Ví dụ 3

numbertree_test_generation.cpp > main()

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

using namespace std;

int main () {
    srand(time(NULL));
    int H = 1 + rand() % 30;
    int k = rand() % H;
    cout<< H << " " ;
    while (k--){
        int r = rand() % 2;
        if (r == 0) cout<< "R" ;
        else cout << "L" ;
    }
    return 0;
}
```

THANKS FOR LISTENING!