# Агроном-любитель

Reads the number of flowers entered and performs a range check to ensure that it is within the bounds (between 1 and 200,000).

Use an array flowers to store the value of the input flower, and perform a range check on each input flower value to ensure that it is within the limited range (between 1 and 10^9).

By looping through the flowers in the array, record the length, start position and end position of the current consecutive flowers.

If the current flower is the same as the previous flower, update the length of the current consecutive flower.

If the current flower is different from the last flower, compare the length of the current continuous flower with the maximum length recorded before, and update the maximum length and the starting position.

Code：

#include <stdio.h>

#include <stdbool.h>

#include <math.h>

int main (){

//check number of flowers

int number;

int number1 =scanf("%d",&number);

if(number>200000||number<1){

return 0;

}//limit (1,200000)

//and check range

int number2= number;

int flowers[number];//create an array

char c;

do{

int type=scanf("%d",&flowers[number-number2]);

if(flowers[number-number2]<1||flowers[number-number2]>pow(10,9)){

return 0;

}//limit (1,10^9)

number2--;

}while((c=getchar())!='\n'&&number2!=0);

//Check the data and put it in the array

int previousPointer = 0;

int pointer = 0;

int previousValue = 0;

int currentValue = 0;

int previousLength = 0;

int currentLength = 0;

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

currentLength++;

pointer = i;

if((currentValue==previousValue)&&(flowers[i]==previousValue)){

if(currentLength-1>previousLength){

previousLength = currentLength-1;

previousPointer = pointer-1;

}

i=i-2;

currentLength = 0;

currentValue = 0;

previousValue = 0;

}else{

previousValue = currentValue;

currentValue = flowers[i];

}

}

int resultEndNumber;

int resultStartNumber;

int resultLength;

if(currentLength<=previousLength){

resultEndNumber = previousPointer+1;

resultLength = previousLength;

resultStartNumber = resultEndNumber - (previousLength-1);

}else{

resultEndNumber = pointer+1;

resultLength = currentLength;

resultStartNumber = resultEndNumber - (currentLength-1);

}

printf("%d %d",resultStartNumber,resultEndNumber);

return 0;

}

# B. Зоопарк Глеба

Define some variables like pointer, pointerForIndex, indexOfAnimals and indexOfIndex to store the indices of the different elements.

Traverse the input string. During the traversal, do the following:

If the stack is empty, push the current character onto the stack.

If the stack is not empty, check the relationship between the current character and the top element of the stack:

If they are not equal and case is satisfied, do the following:

If the current character is a lowercase letter, store the index of the top element in the indexOfAnimals array, and store the index of the top element in the indexOfIndex array.

Pop the top element of the stack and push the current character onto the stack.

If they do not satisfy the case relation, push the current character onto the stack.

After the loop ends, check to see if the stack is empty:

If the stack is not empty, print "Impossible".

If the stack is empty, sort the indexOfIndex array and output "Possible".

Iterates through the sorted indexOfIndex array and uses it as an index to access the indexOfAnimals array, outputting the result.

Code：

#include <iostream>

#include <cstring>

#include <stdbool.h>

#include <algorithm>

#include <stack>

using namespace std;

int main()

{

string str1;

cin>>str1;

stack<pair<int,char>> stk;

int pointer = 1;

int pointerForIndex = 0;

int indexOfAnimals[str1.size()];

int indexOfIndex[str1.size()/2];

for(int i = 0; i < str1.size(); i++){

if(stk.empty()){

if(islower(str1[i])){

stk.emplace(pointer,str1[i]);

pointer++;

}else{

stk.emplace(i,str1[i]);

}

}else{

if(str1[i]!=stk.top().second&&(toupper(str1[i])==stk.top().second||tolower(str1[i])==stk.top().second)){

if(islower(str1[i])){

indexOfAnimals[stk.top().first] = pointer;//stk.top().first is the index of trap in str

indexOfIndex[pointerForIndex] = stk.top().first;

pointerForIndex++;

stk.pop();

pointer++;

}else{

indexOfAnimals[i] = stk.top().first;

indexOfIndex[pointerForIndex] = i;

pointerForIndex++;

stk.pop();

}

}else{

if(islower(str1[i])){

stk.emplace(pointer,str1[i]);

pointer++;

}else{

stk.emplace(i,str1[i]);

}

}

}

}

if(!stk.empty()){

cout<<"Impossible";

}else{

sort(indexOfIndex,indexOfIndex+str1.size()/2);

cout<<"Possible"<<endl;

for(int i = 0; i < str1.size()/2; i++){

cout<<indexOfAnimals[indexOfIndex[i]]<<" ";

}

}

cout<<endl;

return 0;

}

# C. Конфигурационный файл

The program reads input line by line using getline(cin, s).

If the input is an opening brace '{', a new empty vector is added to the remove vector. This indicates the start of a new block of code.

If the input is a closing brace '}', the variables stored in the innermost vector of remove are removed from maps. If a variable's stack is empty or contains only one element after removal, it is erased from maps.

Otherwise, the input is treated as an assignment statement or a variable reference:

The input is split at the '=' sign to separate the variable name and the value/reference.

If the value is a number, it is converted to a long integer and pushed onto the stack of the corresponding variable in maps.

If the value is a variable reference, the top value from the referenced variable's stack is retrieved, pushed onto the stack of the current variable, and printed.

Code：

#include <iostream>

#include <map>

#include <stack>

#include <vector>

using namespace std;//命名空间

bool isNumber(const string str)

{

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

if(i==0){

if(str[i]!='-'&&!isdigit(str[i])){//是否为数字

return false;

}

}else{

if(!isdigit(str[i])){

return false;

}

}

}

return true;

}

int main(){

map<string,stack<long>> maps;//map储存数据的两个部分

vector<vector<string>> remove;//动态数组

vector<string> start;

remove.emplace\_back(start);

string s;

while(getline(cin,s)){//读取输入行，终止符是s

if(!s.compare("{")){

vector<string> new\_remove;

remove.push\_back(new\_remove);

}else if(!s.compare("}")){

for(auto i: remove.back()){

if(maps.find(i)->second.size()<=1){

maps.erase(i);

}else{

maps[i].pop();

}

}

remove.pop\_back();

}else{

size\_t position = s.find("=",0);

string sub11 = s.substr(0,position);

string sub22 = s.substr(position+1,s.size()-position-1);//a = 1

if(isNumber(sub22)){

maps[sub11].emplace(stol(sub22));

remove.back().emplace\_back(sub11);

}//a=b

else{

remove.back().emplace\_back(sub11);

if(!maps[sub22].empty()){

long result =maps[sub22].top();

maps[sub11].emplace(result);

cout<<result<<endl;

}else{

maps[sub11].emplace(0);

cout<<0<<endl;

}

}

}

}

return 0;

}

# D. Профессор Хаос

In each iteration, the number of viruses is updated based on the growth and experimentation.

The number of viruses is multiplied by b (as each virus generates b new viruses).

The number of experimented viruses, c, is subtracted from the total.

If the updated number of viruses is less than or equal to 0, it means all viruses have been experimented on, so the loop is terminated.

If the updated number of viruses is greater than or equal to the capacity of the dish, it is capped at d to match the dish's capacity.

If the updated number of viruses is equal to the previous value, it means the number of viruses won't change further, so the loop is terminated.

Code：

#include <iostream>

using namespace std;

int main()

{

int a;//开始有多少个病毒

int b;//一个病毒可以生成多少个病毒

int c;//用于做实验的病毒

int d;//培养皿的容量

int k;//天数

cin >>a>>b>>c>>d>>k;

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

a=a\*b-c;

if(a<=0){//当数量小于实验数量

a=0;

break;

}else if(a>=d&&a\*b-c>0){//当数量大于容器数量

a=d;

break;

}else if(a\*b-c==a){//a的数量不会变化

break;

}

}

cout<<a;

}