考试说明：

1. 机考总共是3道题，两个一星题和一个二星题目，100+100+200，总分400分。考100分及以上通过。
2. 机考主要是围绕编程基础、数组、字符串、树、循环一类的题目，可以在牛客网、力扣上多刷题，牛客网需要手动写输入输出，考试是在牛客网上进行。

https://www.nowcoder.com/ta/huawei/这个是牛客网练题链接，牛客网上搜索题目：牛客网->题库->在线编程->华为机试在线训练；

leetcode编程题全集：https:leetcode-cn.com/problemset/all/

（需要练习中等以上难度的题目）

1. 考试的时候要提前下载好最新版本的google浏览器，方便打开屏幕监控；手机提前充好电，防止考试过程中手机断电造成监控程序断开。考前在本地安装好调试环境，并提前准备好1个空白界面，方便考试的过程中切出调试代码，代码调试通过后可以直接copy到考试窗口。建议在考试窗口再进行一次调试。可以多次提交。

题目1（100分）

标题：磁盘容量排序 | 时间限制：1秒 | 内存限制：262144K | 语言限制：不限

磁盘的容量单位常用的有M，G，T这三个等级，它们之间的换算关系为1T = 1024G，1G = 1024M，现在给定n块磁盘的容量，请对它们按从小到大的顺序进行稳定排序，例如给定5块盘的容量，1T，20M，3G，10G6T，3M12G9M排序后的结果为20M，3G，3M12G9M，1T，10G6T。注意单位可以重复出现，上述3M12G9M表示的容量即为3M+12G+9M，和12M12G相等。

输入描述:

输入第一行包含一个整数n(2 <= n <= 100)，表示磁盘的个数，接下的n行，每行一个字符串(长度大于2，小于30)，表示磁盘的容量，由一个或多个格式为mv的子串组成，其中m表示容量大小，v表示容量单位，例如20M，1T，30G，10G6T，3M12G9M。

磁盘容量m的范围为1到1024的正整数，容量单位v的范围只包含题目中提到的M，G，T三种，换算关系如题目描述。

输出描述:

输出n行，表示n块磁盘容量排序后的结果。

示例1

输入

3

1G

2G

1024M

输出

1G

1024M

2G

说明

1G和1024M容量相等，稳定排序要求保留它们原来的相对位置，故1G在1024M之前

示例2

输入

3

2G4M

3M2G

1T

输出

3M2G

2G4M

1T

说明

1T的容量大于2G4M，2G4M的容量大于3M2G

参考代码：

#include <iostream>

#include <vector>

#include <string>

using namespace::std;

class distNode {

public:

int sizeM;

string org;

distNode \*next;

distNode(string src) {

sizeM = 0;

next = nullptr;

org = src;

int buff = 0;

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

if (src[i] == 'M') {

sizeM += buff;

buff = 0;

}

else if(src[i] == 'G') {

sizeM += buff \* 1024;

buff = 0;

}

else if (src[i] == 'T') {

sizeM += buff \* 1024 \* 1024;

buff = 0;

}

else {

buff \*= 10;

buff += src[i] - '0';

}

}

}

};

vector<string> sort(vector<string> li) {

vector<string> result;

distNode \*Head = new distNode("0");

for (auto s : li) {

distNode \*node = new distNode(s);

distNode \*p = Head;

bool inserted = false;

while (p->next) {

if (p->next->sizeM > node->sizeM) {

//开始插入

node->next = p->next;

p->next = node;

inserted = true;

break;

}

p = p->next;

}

if (!inserted) {

//放在链表尾部

p->next = node;

}

}

distNode \*tail = Head->next;

while (tail) {

result.push\_back(tail->org);

tail = tail->next;

}

return result;

}

int main()

{

int diskNum;

cin >> diskNum;

vector<string> vs;

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

string s;

cin >> s;

vs.push\_back(s);

}

auto result = sort(vs);

for (auto r : result) {

cout << r << endl;

}

}

题目2（100分）

标题：字符串筛选排序 | 时间限制：1秒 | 内存限制：32768K | 语言限制：不限

输入一个由n个大小写字母组成的字符串，按照Ascii码值从小到大的排序规则，查找字符串中第k个最小ascii码值的字母（k>=1），输出该字母所在字符串的位置索引(字符串的第一个字符位置索引为0）。

k如果大于字符串长度，则输出最大ascii值的字母所在字符串的位置索引，如果有重复的字母，则输出字母的最小位置索引。

输入描述:

第一行输入一个由大小写字母组成的字符串

第二行输入k，k必须大于0，k可以大于输入字符串的长度

输出描述:

输出字符串中第k个最小ascii码值的字母所在字符串的位置索引。k如果大于字符串长度，则输出最大ascii值的字母所在字符串的位置索引，如果第k个最小ascii码值的字母存在重复，则输出该字母的最小位置索引。

示例1

输入

AbCdeFG

3

输出

5

说明

根据ascii码值排序，第3个最小ascii码值的字母为F，F在字符串中的位置索引为5（0为字符串的第一个字母位置索引）

示例2

输入

fAdDAkBbBq

4

输出

6

说明

根据ascii码值排序，前4个字母为AABB ，由于B重复，则只取B的（第一个）最小位置索引6 ，而不是第二个B的位置索引8

参考代码：

#include <iostream>

#include <map>

#include <string>

#include <vector>

using namespace std;

int main() {

map<char, vector<int>> hashMap;

string buf, kStr;

getline(cin, buf);

getline(cin, kStr);

int k = stoi(kStr);

int length = buf.size();

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

char cur = buf[i];

hashMap[cur].emplace\_back(i);

}

if (k >= length) {

auto iter = hashMap.rbegin();

cout << \*(iter->second.begin()) << endl;

return 0;

} else {

int cnt = 0;

for (auto it : hashMap) {

int curSize = it.second.size();

int tmp = curSize + cnt;

if (tmp > k) {

std::cout << \*(it.second.begin()) << endl;

return 0;

}

cnt = tmp;

continue;

}

}

return 0;

}

题目3:（200分）

标题：二叉树的广度优先遍历 | 时间限制：1秒 | 内存限制：262144K | 语言限制：不限

有一棵二叉树，每个节点由一个大写字母标识(最多26个节点）。现有两组字母，分别表示后序遍历（左孩子->右孩子->父节点）和中序遍历（左孩子->父节点->右孩子）的结果，请输出层次遍历的结果。

输入描述:

输入为两个字符串，分别是二叉树的后续遍历和中序遍历结果。

输出描述:

输出二叉树的层次遍历结果。

示例1

输入

CBEFDA CBAEDF

输出

ABDCEF

说明

二叉树为：

A

/ \

B D

/ / \

C E F

参考代码：

#include <iostream>

#include <string>

#include <vector>

using namespace std;

struct BinarytreeNode {

char data;

BinarytreeNode \*left;

BinarytreeNode \*right;

BinarytreeNode(char value='\n', BinarytreeNode \*l=NULL,BinarytreeNode \*r=NULL):data(value),left(l),right(r)

{}

};

BinarytreeNode\* Createtree(string in, string post, int n)

{

if (n == 0) return NULL;

int k = n - 1;

while (post[n-1] != in[k]) { k--; }

BinarytreeNode \*rootelement = new BinarytreeNode(post[n - 1]);

rootelement->left = Createtree(in,post,k);

rootelement->right = Createtree(in.substr(k + 1, -1), post.substr(k, -1), n-k-1);

return rootelement;

}

string printNode(BinarytreeNode root) {

vector<BinarytreeNode> t;

t.push\_back(root);

string result;

while (t.size()) {

result += t[0].data;

if (t[0].left) {

t.push\_back(\*t[0].left);

}

if (t[0].right) {

t.push\_back(\*t[0].right);

}

t.erase(t.begin(), t.begin() + 1);

}

return result;

}

int main() {

string post;

cin >> post;

string in;

cin >> in;

int n = in.size();

BinarytreeNode\* tree = Createtree(in, post, n);

cout << printNode(\*tree) << endl;

return 0;

}

题目4（100分）

标题：寻找身高相近的小朋友 | 时间限制：1秒 | 内存限制：262144K | 语言限制：不限

小明今年升学到小学一年级，来到新班级后发现其他小朋友们身高参差不齐，然后就想基于各小朋友和自己的身高差对他们进行排序，请帮他实现排序。

输入描述:

第一行为正整数H和N，0<H<200，为小明的身高，0<N<50，为新班级其他小朋友个数。

第二行为N个正整数H1-HN，分别是其他小朋友的身高，取值范围0<Hi<200（1<=i<=N），且N个正整数各不相同。

输出描述:

输出排序结果，各正整数以空格分割。和小明身高差绝对值最小的小朋友排在前面，和小明身高差绝对值最大的小朋友排在最后，如果两个小朋友和小明身高差一样，则个子较小的小朋友排在前面。

**示例1**

输入

100 10

95 96 97 98 99 101 102 103 104 105

输出

99 101 98 102 97 103 96 104 95 105

说明

小明身高100，班级学生10个，身高分别为95 96 97 98 99 101 102 103 104 105，按身高差排序后结果为：99 101 98 102 97 103 96 104 95 105。

题目5（100分）

标题：喊7的次数重排 | 时间限制：1秒 | 内存限制：262144K | 语言限制：不限

喊7是一个传统的聚会游戏，N个人围成一圈，按顺时针从1到N编号。编号为1的人从1开始喊数，下一个人喊的数字为上一个人的数字加1，但是当将要喊出来的数字是7的倍数或者数字本身含有7的话，不能把这个数字直接喊出来，而是要喊"过"。假定玩这个游戏的N个人都没有失误地在正确的时机喊了"过"，当喊到数字K时，可以统计每个人喊"过"的次数。

现给定一个长度为N的数组，存储了打乱顺序的每个人喊"过"的次数，请把它还原成正确的顺序，即数组的第i个元素存储编号i的人喊"过"的次数。

输入描述:

输入为一行，为空格分隔的喊"过"的次数，注意K并不提供，K不超过200，而数字的个数即为N。

输出描述:

输出为一行，为顺序正确的喊"过"的次数，也由空格分隔。

**示例1**

输入

0 1 0

输出

1 0 0

说明

一共只有一次喊"过"，那只会发生在需要喊7时，按顺序，编号为1的人会遇到7，故输出1 0 0。注意，结束时的K不一定是7，也可以是8、9等，喊过的次数都是1 0 0。

**示例2**

输入

0 0 0 2 1

输出

0 2 0 1 0

说明

一共有三次喊"过"，发生在7 14 17，按顺序，编号为2的人会遇到7 17，编号为4的人会遇到14，故输出0 2 0 1 0。

练习题6（200分）

标题：目录删除 | 时间限制：1秒 | 内存限制：262144K | 语言限制：不限

某文件系统中有N个目录，每个目录都一个独一无二的ID。每个目录只有一个父目录，但每个父目录下可以有零个或者多个子目录，目录结构呈树状结构。

假设，根目录的ID为0，且根目录没有父目录，其他所有目录的ID用唯一的正整数表示，并统一编号。

现给定目录ID和其父目录ID的对应父子关系表[子目录ID，父目录ID]，以及一个待删除的目录ID，请计算并返回一个ID序列，表示因为删除指定目录后剩下的所有目录，返回的ID序列以递增序输出。

注意：

1、被删除的目录或文件编号一定在输入的ID序列中；

2、当一个目录删除时，它所有的子目录都会被删除。

输入描述:

输入的第一行为父子关系表的长度m；接下来的m行为m个父子关系对；最后一行为待删除的ID。序列中的元素以空格分割，参见样例。

输出描述:

输出一个序列，表示因为删除指定目录后，剩余的目录ID。

**示例1**

输入

5

8 6

10 8

6 0

20 8

2 6

8

输出

2 6

说明

目录结构如下所示：

     6

   /   \  
 2     8  
      /  \  
     10  20  
  
删除目录8，同时它的子目录10也被删除，剩余2和6两个目录。

练习题6（100分）

标题：计算最大乘积 | 时间限制：1秒 | 内存限制：32768K | 语言限制：不限

给定一个元素类型为小写字符串的数组，请计算两个没有相同字符的元素 长度乘积的最大值，如果没有符合条件的两个元素，返回0。

**输入描述:**

输入为一个半角逗号分隔的小写字符串的数组，2 <= 数组长度<=100，0 < 字符串长度<= 50。

**输出描述:**

两个没有相同字符的元素 长度乘积的最大值。

**示例1**

输入

iwdvpbn,hk,iuop,iikd,kadgpf

输出

14

说明

数组中有5个元素。

iwdvpbn与hk无相同的字符，满足条件，iwdvpbn的长度为7，hk的长度为2，乘积为14（7\*2）。

iwdvpbn与iuop、iikd、kadgpf均有相同的字符，不满足条件。

iuop与iikd、kadgpf均有相同的字符，不满足条件。

iikd与kadgpf有相同的字符，不满足条件。

因此，输出为14。

参考代码：

#include <iostream>

#include <vector>

#include <string>

using namespace std;

void tok(vector<string> &arr, string str, string pattern) {

string strs = str + pattern;

int pos = strs.find(pattern);

int size = strs.size();

while (pos != string::npos) {

string t = strs.substr(0, pos);

arr.push\_back(t);

strs = strs.substr(pos + 1, size);

pos = strs.find(pattern);

}

}

void packing(vector<int> &tmp, string s) {

for (auto i : s) {

tmp[i - 'a'] = 1;

}

}

int calc(vector<string> &arr){

int res = 0;

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

vector<int> tmp(26,0);

packing(tmp, arr[i]);

for (int j = i + 1; j < arr.size(); ++j) {

int flag = 0;

for (int k = 0; k < arr[j].size(); ++k) {

if (tmp[arr[j][k]-'a'] == 1) {

flag = 1;

break;

}

}

if (flag == 0) {

int t = arr[i].size() \* arr[j].size();

res < t ? (res = t) : 0;

}

}

}

return res;

}

int main() {

vector<string> arr;

string input;

cin >> input;

tok(arr, input, ",");

cout << calc(arr) << endl;

return 0;

}

练习题8（100分）

标题：拼接URL | 时间限制：1秒 | 内存限制：32768K | 语言限制：不限

给定一个URL前缀和URL后缀，通过","分割，需要将其连接为一个完整的URL，如果前缀结尾和后缀开头都没有“/”，需自动补上“/”连接符，如果前缀结尾和后缀开头都为“/”，需自动去重。

约束：不用考虑前后缀URL不合法情况。

输入描述:

URL前缀（一个长度小于100的字符串),URL后缀（一个长度小于100的字符串）。

输出描述:

拼接后的URL。

**示例1**

输入

/acm,/bb

输出

/acm/bb

**示例2**

输入

/abc/,/bcd

输出

/abc/bcd

参考代码：

#include <iostream>

#include <string>

#include <regex>

using namespace std;

string get\_value(const string& str) {

string s = regex\_replace(str, regex(","), "/");

s = regex\_replace(s, regex("/+"), "/");

return s;

}

int main() {

string str;

cin >> str;

if (str.find(',') != string::npos)

cout << get\_value(str);

else {

cout << "\n";

}

return 0;

}

练习题9（200分）

标题：解压报文 | 时间限制：1秒 | 内存限制：262144K | 语言限制：不限

为了提升数据传输的效率，会对传输的报文进行压缩处理。输入一个压缩后的报文，请返回它解压后的原始报文。

压缩规则：n[str]，表示方括号内部的 str 正好重复 n 次。注意 n 为正整数（0 < n <= 100），str只包含小写英文字母，不考虑异常情况。

输入描述:

输入压缩后的报文：

1）不考虑无效的输入，报文没有额外的空格，方括号总是符合格式要求的；

2）原始报文不包含数字，所有的数字只表示重复的次数 n ，例如不会出现像 5b 或 3[8] 的输入；

输出描述:

解压后的原始报文

注：

1. 原始报文长度不会超过1000，不考虑异常的情况

**示例1**

输入

3[k]2[mn]

输出

kkkmnmn

说明

k 重复3次，mn 重复2次，最终得到 kkkmnmn

**示例2**

输入

3[m2[c]]

输出

mccmccmcc

说明

m2[c] 解压缩后为 mcc，重复三次为 mccmccmcc

参考代码：

#include <iostream>

#include <vector>

#include <regex>

#include <string>

using namespace std;

string operator\*(const string &str, int n)

{

string ss;

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

ss.append(str);

}

return ss;

}

string calc(const string &str, vector<string> &arr)

{

string result;

while (!arr.empty()) {

string back = arr.back();

arr.pop\_back();

if (isdigit(back.front())) {

result = result \* stoi(back);

break;

}

else {

result = back + result;

}

}

return result;

}

int main()

{

string str;

cin >> str;

regex pp("(\\d+)|(\\[)|(\\])|([a-z]+)");

string::const\_iterator iter = str.cbegin();

smatch math;

vector<string> arr;

while (regex\_search(iter, str.cend(), math, pp)) {

iter = math[0].second;

string symbol;

for (size\_t i = 1; i < math.size(); ++i) {

if (math[i].length() > 0) {

symbol = math[i];

break;

}

}

if (symbol == "]") {

arr.push\_back(calc(symbol, arr));

}

else if (symbol != "["){

arr.push\_back(symbol);

}

}

str.clear();

for (const auto &i : arr) {

str.append(i);

}

cout << str << endl;

return 0;

}

练习题10（100分）

标题：数组组成的最小数字 | 时间限制：1秒 | 内存限制：262144K | 语言限制：不限

给定一个整型数组，请从该数组中选择3个元素组成最小数字并输出（如果数组长度小于3，则选择数组中所有元素来组成最小数字）。

**输入描述:**

一行用半角逗号分割的字符串记录的整型数组，0 < 数组长度 <= 100，0 < 整数的取值范围 <= 10000。

**输出描述:**

由3个元素组成的最小数字，如果数组长度小于3，则选择数组中所有元素来组成最小数字。

**示例1**

输入

21,30,62,5,31

输出

21305

说明

数组长度超过3，需要选3个元素组成最小数字，21305由21,30,5三个元素组成的数字，为所有组合中最小的数字

**示例2**

输入

5,21

输出

215

说明

数组长度小于3， 选择所有元素来主城最小值，215为最小值。

练习题11（100分）

标题：连续字母长度 | 时间限制：1秒 | 内存限制：262144K | 语言限制：不限

给定一个字符串，只包含大写字母，求在包含同一字母的子串中，长度第 k 长的子串的长度，相同字母只取最长的那个子串。

**输入描述:**

第一行有一个子串(1<长度<=100)，只包含大写字母。

第二行为 k的值

**输出描述:**

输出连续出现次数第k多的字母的次数。

**示例1**

输入

AAAAHHHBBCDHHHH

3

输出

2

说明

同一字母连续出现的最多的是A和H，四次；第二多的是H，3次，但是H已经存在4个连续的，故不考虑；下个最长子串是BB，所以最终答案应该输出2。

**示例2**

输入

AABAAA

2

输出

1

说明

同一字母连续出现的最多的是A，三次；第二多的还是A，两次，但A已经存在最大连续次数三次，故不考虑；下个最长子串是B，所以输出1

练习题12（200分）

标题：可以组成网络的服务器 | 时间限制：1秒 | 内存限制：262144K | 语言限制：不限

在一个机房中，服务器的位置标识在 n\*m 的整数矩阵网格中，1 表示单元格上有服务器，0 表示没有。如果两台服务器位于同一行或者同一列中紧邻的位置，则认为它们之间可以组成一个局域网。

请你统计机房中最大的局域网包含的服务器个数。

**输入描述:**

第一行输入两个正整数，n和m，0<n,m<=100  
之后为n\*m的二维数组，代表服务器信息

**输出描述:**

最大局域网包含的服务器个数。

**示例1**

输入

2 2

1 0

1 1

输出

3

说明

[0][0]、[1][0]、[1][1]三台服务器相互连接，可以组成局域网