**2016年秋季校招（研发类）招聘笔试题**

1. 代码纠错。下列代码实现了升序数组的二分查找，找出其中存在的错误并改正。请用红色字体高亮改正部分

#include <stdio.h>

int BiSearch(int l , int r , int tmp , int a[]){

if (l >= r && a[l] != tmp){

return -1;

}

int mid = (l + r)/2;

if (a[mid] == tmp){

return mid;

}

if (a[mid] < tmp){

return BiSearch(mid , r , tmp , a);

}

return BiSearch(l , mid , tmp , a);

}

int main(int argc , char \*argv[]){

int a[10] = {0,1,2,3,4,5,6,7,8,9};

int res = 0 , tmp;

scanf("%d",&tmp);

res = BiSearch(0 , 9 , tmp , a);

printf("%d",res);

return 0;

}

下边两道编程题，可以任选C++或Java完成

2. 快乐数

快乐数定义如下：从任意一个正整数开始，计算这个数字每一位的数字的平方的和，得到一个数字；然后重复这一过程，直到这个数字收敛到1；或者陷入到一个不包含1的循环。如果能收敛到1的，就定义为快乐数。

例如，19是快乐数：

* 12 + 92 = 82
* 82 + 22 = 68
* 62 + 82 = 100
* 12 + 02 + 02 = 1

输出一个数字，判断是否是快乐数。

任选Java或C++

C++实现：

#include <iostream>

#include<vector>

#include<algorithm>

using namespace std;

int getSquareSum(int n)

{

int val = 0;

while (n) {

int rem = n%10;

val += rem\*rem;

n = n / 10;

}

return val;

}

class solution{

public:

bool isHappy(int n) {

vector<int> result;

int val = getSquareSum(n);

while (val != 1) {

if (std::find(result.begin(), result.end(), val) != result.end()) {

return false;

}

result.push\_back(val);

val = getSquareSum(val);

}

return true;

}

};

int main()

{

int m;

cin>>m;

solution s;

if(s.isHappy(m)) cout<<"YES";

else cout<<"NO";

return 0;

}

3.二叉树路径和

给定一棵二叉树，二叉树权值为0-9，得出所有根节点到叶节点的路径和。如下图的二叉树，路径和n=137+15=152

1

/ \

3 5

\

7

任选Java或C++

#include <iostream>

#include<math.h>

#define M 100000

using namespace std;

//二叉树定义

struct Node {

int value;

Node \*left;

Node \*right;

}Tree[M];

int loc;

Node\*create(){

Tree[loc].left = Tree[loc].right = NULL;

return &Tree[loc++];

}

int \* arr1,arr2;

Node\* build(int s1,int s2,int e1,int e2)

{

Node \* ret = create();

ret->value = arr1[s1];

int rootidx;

for (int i= s2;i<=e2;i++)

{

if(arr2[i]==arr1[s1]){

rootidx = 1;

break;

}

}

if(rootidx!=s2){

ret->left = build(s1+1,s1+(rootidx-s2),s2,rootidx-1);

}

if(rootidx!=e2){

ret->right = build(s1+(rootidx-s2)+1,e1,rootidx,e2);

}

return ret;

}

class Solution {

public:

int sumNumbers(Node\* root) {

int pathnum[M] = {0};

int sum = 0;

if(!root) return 0;

if(!root->left&&!root->right)

{

int i;

for(i=0;i<loc;i++)

pathnum[i]=root->value;

sum+= pathnum[i]\*pow(10,i);

}

else {

sumNumbers(root->left);

sumNumbers(root->right);

}

return sum;

}

};

int main()

{

int i=0;

while(cin>>!='\n'))

{

cin>>arr1[i];

i++;

}

while(scanf(%d!='\n'))

{

cin>>arr2[i]);

i++;

}

loc = 0;

int L1 = strlen(arr1);

int L2 = strlen(arr2);

Node \* T = build(0,L1-1,0,L2-1,arr1,arr2);

Solution s;

int res = s.sumNumbers(T);

cout << res << endl;

return 0;

}