初赛模拟试卷二

初赛选择模拟练习1

1、第一个给计算机写程序的人是（ ）。

A.Alan Mathison Turing B.Ada Lovelace C. John von Neumann D.John Mc-Carthy

2、下列无符号数中，最小的数是（ ）

A.　 B.　　 C.　 　D.

3、现有一张分辨率为 2048×1024 像素的 16色真彩色图像。请问要存储这张图像，需要多大的存储空间？（ ）。

A.1MB B.2MB　 C.4MB 　D.

4、断电后计算机信息依然存在的部件为( )。

A.寄存器  B.RAM存储器    C.ROM存储        D.运算器

1. 以下排序算法中，（ ）是不稳定的算法。
2. 插入排序 B. 归并排序 C.冒泡排序 D.选择排序

6、有2×n的一个长方形方格，用一个1×2的骨牌铺满方格。例如n=3时，为2×3方格。此时用一个1×2的骨牌铺满方格，共有3种铺法：

1. 若当n = 5时，那么铺法总数为（ ）。

A.5 B.6 C.7 D.8

1. XX和爸爸妈妈三个人一起想动手做三道菜。XX负责洗菜、爸爸负责切菜、妈妈负责炒菜。假设做每道菜的顺序都是：先洗菜 10 分钟，然后切菜 10 分钟，最后炒菜 10 分钟。那么做一道菜需要 30 分钟。注意：两道不同的菜的相同步骤不可以同时进行。例如第一道菜和第二道的菜不能同时洗，也不能同时切。那么做完三道菜的最短时间需要（ ）分钟。

A.40 B.50 C.60 D.90

8、G 是一个非连通简单无向图，共有 28 条边，则该图至少有（ ）个顶点。

A. 10 B. 9 C. 8  D. 7

9、开放式系统互联通信参考模型（OSI模型）是一个试图使各种计算机在世界范围内互连为网络的标准框架，那么OSI模型一共有（ ）层。

A.3 B.5 C.7 D.9

10、设A是n个实数的数组，考虑下面的递归算法：

NOP（x, A[1 .. n]）

1.Mid ← (1 + n) >> 1

2.If a[mid]=x then mid

3.If a[mid]>x

4. then temp ← NOP（x, A[1 .. mid-1]）

5. else temp ← NOP（x, A[mid+1 .. n]）

请问算法NOP中要求a数组满足（ ）

A.呈单调递增 B.呈单调递减

C.呈正态分布 D.随意均可

11、欧拉图G是指可以构成一个闭回路的图，且图G的每一条边恰好在这个闭回路上出现一次（即一笔画成）。在以下各个描述中, 不一定是欧拉图的是（ ）。

A. 图G中没有度为奇数的顶点

B. 包括欧拉环游的图(欧拉环游是指通过图中每边恰好一次的闭路径)

C. 包括欧拉闭迹的图(欧拉迹是指通过途中每边恰好一次的路径)

D. 存在一条回路, 通过每个顶点恰好一次

1. 如果一颗二叉树的前序遍历是ABCDEFG，后序遍历是CBFEGDA，那么根结点的左子树的结点个数可能是（   ）。

A.0 B.2 C.4 D.6

1. 如果现在有1个a，2个b，3个c，排成一排，则一共有（ ）种排列方法

A.60 B.120 C.360 D.720

14、现在有N 位小朋友在操场围成一圈，将这 N 位小朋友按顺时针方向从1 到N 编号，然后，从第一个人起，每隔一个人让下一个人离开操场，显然，第一轮过后，具有偶数编号的人都离开了操场。依次做下去，直到操场只剩下一位小朋友，记这个人的编号为F(N) ，例如，F(5)=3 ，F(10)=5 ，等等。则 F(400) =（ ）。

（提示：对进行分析，其中）。

A.144 B.145 C.288 D.289

15、NOIP 初赛中，选手可以带入考场的有（ ）。

A.自己的初赛资料 B.计算器 C. 准考证 D.草稿纸

二、阅读程序（程序输入不超过数组或字符串定义的范围；判断题正确填 √ ，错误填 × ；除特 殊说明外，判断题 1.5 分，选择题 3 分，共计 40 分）

（1）

**#include <iostream>**

**using namespace std;**

**void fun(char \*a, char \*b, char \*c) {**

**a = b;**

**b = c;**

**(\*a)++;**

**}**

**int main() {**

**char c1, c2, c3, \*p1, \*p2, \*p3;**

**cin >> c1 >> c2 >> c3;**

**p1 = &c1;**

**p2 = &c2;**

**p3 = &c3;**

**fun(p1, p2, p3);**

**cout << c1 << c2 << c3 << endl;**

**return 0;**

**}完成下面的判断题和单选题：**

判断题：

16、若将程序中所有对于变量的类型声明均从char改成string，不会影响程序的运行。（ ）

17、若将第10行中” cin >> c1 >> c2 >> c3;”改成“scanf("%c%c%c",&c1, &c2, &c3);”，则不会影响程序运行的结果。 （ ）

18、若将第5行中” (\*a)++;”改成“（a）++”，则不会影响程序运行的结果。（ ）

19、当输入为“NaO”时，则输出结果为“NbO”。（ ）

单选题：

20、当输入的结果为“c9p”, 那么输出结果为 （ ）

A．c:p

B．c9q

C．c0p

D．以上结果都不对

21、 这段代码的含义为（ ）。

A. 将第二个字符指针的地址增加1

B. 将第三个字符指针的地址增加1

C. 将第二个字符指针所指向的字符Ascii增加1

D. 将第三个字符指针所指向的字符Ascii增加1

（2）

#include <iostream>

using namespace std;

string s;

long long magic(int l, int r) {

long long sum = 0;

for (int i = l; i <= r; ++i) {

sum = sum \* 4 + s[i] - 'a' + 1;

}

return sum;

}

int main() {

cin >> s;

int len = s.length();

int ans = 0;

for (int l1 = 0; l1 < len; ++l1) {

for (int r1 = l1; r1 < len; ++r1) {

bool ff = true;

for (int l2 = 0; l2 < len; ++l2) {

for (int r2 = l2; r2 < len; ++r2) {

if (magic(l1, r1) == magic(l2, r2)

&& (l1 != l2 || r1 != r2))

ff = false;

}

}

if (ff) {

ans += 1;

}

}

}

cout << ans << endl;

return 0;

}

判断题：

22、若输入的数据中存在4种不同的字符，则会对程序结果产生影响。（ ）

23、将第20行的“for (int l2 = 0; l2 < len; ++l2)”改成“for (int l2 = l1 + 1; l2 < len; ++l2)”，则不会影响程序的运行结果。（ ）

24、将第23行的“(l1 != l2 || r1 != r2)“ 改成 ”(l1 != l2 && r1 != r2)”，不会影响程序的运行结果。（ ）

选择题：

25、若输入的字符串为“abacaba”，则输出结果为（ ）

A．14 B. 15 C.16 D.18

26、magic的函数采用的算法是（ ）

A．倍增算法

B．贪心算法

C．哈希算法

D．搜索算法

27、若输入的数据中包含4种不同的字符，输入字符串为“baiic”，那么输出数据应该是（ ）

A．8 B.10 C.12 D.15

（3）

#include <iostream>

#include<bits/stdc++.h>

using namespace std;

int lps(string seq, int i, int j)

{

int len1, len2;

if (i == j)

return 1;

if (i > j)

return 0;

if (seq[i] == seq[j])

return lps(seq, i + 1, j - 1) + 2;

len1 = lps(seq, i, j - 1);

len2 = lps(seq, i + 1, j);

if (len1 > len2)

return len1;

return len2;

}

int main()

{

string seq;

getline(cin, seq);

int n = seq.size();

cout << lps(seq, 0, n - 1) << endl;

return 0;

}

28、该程序的输出结果一定为奇数。（ ）

29、若将第15行的“len1 > len2” 改为“len1 < len2”,那么输出结果一定为1。（ ）

30、当输入的字符串为“acmerandacm”，则输出结果为5.（ ）

31、该程序最好的时间复杂度为（ ）

A、O（n）

B、O（n^2）

C、O(nlogn)

D、O(2^n)

32、若输入数据为“abcdbcabcd，则输出结果为（ ）

A、1 B、3 C、5 D、7

33、当去掉第11行和第12行的代码后，程序的时间复杂度为（ ）

A、O（n）

B、O（n^2）

C、O(nlogn)

D、O(2^n)

## 三、完善程序（单选题，每小题 3 分，共计 30 分）

（1）

给定两个正整数 a和 b，求在 [a,b]中的所有整数中，每个数码(digit)各出现了多少次。当然第一眼看到题目大家都会觉得是用暴力解决，但是当我们的a，b满足1<= a , b <= 10^12时，我们就需要采用数位dp来解决这个问题。对于特别大的数据我们可以从数位的角度进行拆分，从个位数到十位数，从十位数到百位数……令f[i]为i位数(算前导零)中每个数出现的次数(其实每个数字一定是相同的，所以只记录一个就行了)

如果我们要求[a,b]的答案我们不妨求出[0,b]的答案减去[0,a - 1]的答案，类似于前缀和的方式。

#include <bits/stdc++.h>

#define rep( i , l , r ) for( int i = (l) ; i <= (r) ; i++ )

#define per( i , r , l ) for( int i = (r) ; i >= (l) ; i-- )

using namespace std;

typedef long long ll;

inline ll \_read(){

register ll x = 0;

register char ch = getchar();

while( ch > '9' || ch < '0' ) ch = getchar();

while( ch >= '0' && ch <= '9' ){

x = x \* 10 + ch - '0';

ch = getchar();

}

return x;

}

const int maxn = 10 + 5;

ll ans[maxn] , f[2 \* maxn];

inline void resolve( ll x , ll pos ){

while( x ) ans[ x % 10 ] += pos , （1）;

}

void solve( ll x , int flg ){

int i , j ;

ll pos , now;

for( i = 1 , pos = 10 ; pos < x ; ++i , pos \*= 10 ){

rep( j , 0 , 9 ) (2)

rep( j , 1 , 9 ) ans[j] += pos / 10 \* flg;

}

now = pos /= 10 ; --i;

while( now < x ){

while( now + pos <= x ){

ll t = now / pos ;

(3)

rep( j , 0 , 9 ) ans[j] += f[i] \* flg;

now += pos;

}

pos /= 10; --i;

}

}

int main(){

int i ;

ll a , b , pos ;

f[1] = 1;

for( i = 2 , pos = 10 ; i <= 12 ; ++i , pos \*= 10 ) f[i] = f[i - 1] \* 10 + pos;

a = \_read() , b = \_read();

solve( b + 1 , 1 );(4)

rep( j , 0 , 9 ) printf("%lld%c" , ans[j] , j == 9 ? '\n' : ' ');

return 0;

}

34、（ ）

A．x /= 10 B. x %= 10 C.x >> 1 D. x << 1

35、（ ）

A. ans[j] += f[i] \* 9;

B. ans[j] += f[i - 1] \* 9;

C. ans[j] += f[i] \* 9 \* flg;

D. ans[j] += f[i - 1] \* 9 \* flg;

36、（ ）

A．resolve( now , pos \* flg);

B．resolve( now , pos);

C．resolve( t , pos);

D．resolve( t , pos \* flg );

37、（）

A．solve( a - 1 , -1 );

B．solve( a , -1 );

C．solve( a + 1, -1 );

D．solve(b – a + 1 , -1 );

(2)

学会了简单的加减乘除的高精度，那么对于开方运算的高精度也是必须要掌握的。对于一个有理数开根之后可能得到一个无理数，所以这项工作是有较大难度的。现在要做的只是这项工作的第一步：只对自然数进行开整数次根，求出它的一个非负根，并且不考虑结果的小数部分，只要求把结果截断取整即可。程序中给定需要开根的次数以及被开根的整数，要求计算出它的非负根取整后的结果。

#include<bits/stdc++.h>

#include<iostream>

#include<cstdio>

#define ll long long

using namespace std;

const ll N = 20010;

ll m;

struct bignum{

ll len,num[N];

bignum(){len = 0; memset(num,0,sizeof(num));}

}n,one,zero,ans;

inline bignum read(){ //输入

bignum ans;

string s;

cin>>s;

for (ll i = 0; i < s.size(); i++) (1)

ans.len = s.size();

return ans;

}

inline void write(bignum s){ //输出

for (ll i = s.len; i >= 1; i--) putchar(s.num[i] + 48);

}

inline void operator ==(bignum &a,ll b){

for (; b; b/=10) (2)

}

inline bool operator <=(bignum a,bignum b){

if (a.len > b.len) return 0;

if (a.len < b.len) return 1;

for (ll i = a.len; i >= 1; i--){

if (a.num[i] > b.num[i]) return 0;

if (a.num[i] < b.num[i]) return 1;

}

return 1;

}

inline bool operator >(bignum a,bignum b){

if (a.len > b.len) return 1;

if (a.len < b.len) return 0;

for (ll i = a.len; i >= 1; i--){

if (a.num[i] > b.num[i]) return 1;

if (a.num[i] < b.num[i]) return 0;

}

(3)

}

inline bignum operator +(bignum a,bignum b){

bignum c;

c.len = max(a.len,b.len);

for (ll i = 1; i <= c.len; i++){

c.num[i]+=a.num[i] + b.num[i];

c.num[i+1]+=c.num[i] / 10;

c.num[i]%=10;

}

if (c.num[c.len + 1]) c.len++;

return c;

}

inline bignum operator -(bignum a,bignum b){

bignum c;

c.len = a.len;

for (ll i = 1; i <= c.len; i++){

c.num[i] = a.num[i] - b.num[i];

if (c.num[i] < 0){

c.num[i]+=10;

a.num[i + 1]--;

}

}

while (c.len > 1 && c.num[c.len] == 0) c.len--;

return c;

}

inline bignum operator \*(bignum a,bignum b){

bignum c;

for (ll i = 1; i <= a.len; i++){

for (ll j = 1; j <= b.len; j++){

(4)

c.num[i + j]+=c.num[i + j - 1] / 10;

c.num[i + j - 1]%=10;

}

}

c.len = a.len + b.len;

while(c.len > 1 && c.num[c.len] == 0) c.len--;

return c;

}

inline bignum operator /(bignum a,ll b){

ll sum = 0; bignum c;

for (ll i = a.len; i >= 1; i--){

sum = sum \* 10 + a.num[i];

c.num[i] = sum / b;

sum % =b;

}

c.len = a.len;

while (c.len > 1 && c.num[c.len] == 0) c.len--;

return c;

}

inline bool check(bignum a,ll b){

bignum ans = one;

for (ll i = 1; i <= b; i++){

ans = ans \* a;

if (ans > n) return 0;

}

return 1;

}

int main(){

cin>>m;

n = read();

（5）

zero == 0;

bignum l = one, r = n;

while (l <= r){

bignum mid = (l + r) / 2;

if (check(mid, m)){

ans = mid;

(6)

}

else r = mid - one;

}

write(ans);

return 0;

}

38、（ ）

A、ans [s.size() - i] = s[i] - 48;

B、ans [s.size() - i] = s[i];

C、ans.num[s.size() - i] = s[i];

D、ans.num[s.size() - i] = s[i] - 48;

39、（ ）

A、a.num[++a.len]=b%10;

B、a.num[a.len]=b%10;

C、a.num[a.len++]=b%10;

D、a.num[&a.len]=b%10;

40、（ ）

A、return 1;

B、return 0;

C、空

D、return i > 0;

41、（ ）

A、c.num[i + j]+=a.num[i] \* b.num[j];

B、c.num[i + j]+=a.num[i + 1] \* b.num[j + 1];

C、c.num[i + j - 1]+=a.num[i] \* b.num[j];

D、c.num[i + j - 1]+=a.num[i + 1] \* b.num[j + 1];

42、（ ）

A、one = 1;

B、one == 1;

C、one =（bignum）1;

D、one == “1”;

43、（ ）

A、l = mid + one; B、l = mid; C、l = mid + 1; D、l = mid + zero;