1.[单选题]

当分配给一个进程的页面数增加时，页故障数可能增大也可能变小，下述算法符合这种情况的是

FIFO算法

LRU算法

Clock算法

LFU算法

2. [单选题]

在Linux系统中哪个程序是特定的时间上安排特定的作业或者程序的运行？

vi

Outlook

Scheduler

cron

3. [单选题]

下面代码的输出结果是什么 ( )

#include <iostream>

using namespace std;

class A

{

    public: void virtual f()

    {

        cout << "A" << " ";

    }

};

class B : public A

{

    public: void virtual f()

    {

        cout << "B" << " ";

    }

};

int main(){

    A \*pa = new A();

    pa->f();

    B \*pb=(B \*)pa;

    pb->f();

    delete pa, pb;

    pa=new B();

    pa->f();

    pb=(B \*)pa;

    pb->f();

    return 0;

}

A A B A

A B B A

A A B B

A A A B

4. [单选题]

一个提供NAT服务的路由器在转发一个源IP地址为10.0.0.1、目的IP地址为131.12.1.1的IP分组时，可能重写的IP分组首部字段是

Ⅰ.TTL

Ⅱ.片偏移量

Ⅲ.源IP地址

Ⅳ.目的IP地址

仅Ⅰ

仅I、Ⅱ

仅Ⅰ、Ⅱ、III

Ⅰ、Ⅱ、Ⅲ、Ⅳ

5. [单选题]

sql中，可以用来替换DISTINCT的语句是（ ）

ORDER BY

GROUP BY

DESC

HAVING

6.[单选题]

在32位计算机中，下面输出是多少（ ）

#include <iostream>

using namespace std;

typedef enum

{

Char ,

Short,

Int,

Double,

Float,

}TEST\_TYPE;

int main() {

TEST\_TYPE val;

cout<< sizeof(val)<<endl;

return 0;

}

5

4

8

12

7. [单选题]

以下程序中，当输入6个字符后，容器的当前大小(size)是多少（）

int main()

{

vector<char> vec;

vec.reserve(1024);

string tmp;

cin >> tmp;

string::iterator it = tmp.begin();

for(; it!=tmp.end(); it++)

{

vec.push\_back(\*it);

}

vec.resize(vec.size() +vec.size()/2);

return 0;

}

6

9

1024

1536

8. [单选题]

对于字符串"ABCDADA"的二进制哈夫曼编码有多少位?

11

12

13

14

9. [单选题]

下面关于选择排序说法正确的是()

每扫描一遍数组，需要多次交换

选择排序是稳定的排序方法,因为时间复杂度是固定的O（n^2）

选择排序排序速度一般要比冒泡排序快

空间复杂度为O（1）

10. [编程题] 红和绿

时间限制：1秒

空间限制：32768K

牛牛有一些排成一行的正方形。每个正方形已经被染成红色或者绿色。牛牛现在可以选择任意一个正方形然后用这两种颜色的任意一种进行染色,这个正方形的颜色将会被覆盖。牛牛的目标是在完成染色之后,每个红色R都比每个绿色G距离最左侧近。牛牛想知道他最少需要涂染几个正方形。

如样例所示: s = RGRGR

我们涂染之后变成RRRGG满足要求了,涂染的个数为2,没有比这个更好的涂染方案。

输入描述:

输入包括一个字符串s,字符串s长度length(1 ≤ length ≤ 50),其中只包括'R'或者'G',分别表示红色和绿色。

输出描述:

输出一个整数,表示牛牛最少需要涂染的正方形数量

输入例子1:

RGRGR

输出例子1:

2

11. [编程题] 奶牛编号

时间限制：1秒

空间限制：32768K

牛牛养了n只奶牛,牛牛想给每只奶牛编号,这样就可以轻而易举地分辨它们了。 每个奶牛对于数字都有自己的喜好,第i只奶牛想要一个1和x[i]之间的整数(其中包含1和x[i])。

牛牛需要满足所有奶牛的喜好,请帮助牛牛计算牛牛有多少种给奶牛编号的方法,输出符合要求的编号方法总数。

输入描述:

输入包括两行,第一行一个整数n(1 ≤ n ≤ 50),表示奶牛的数量 第二行为n个整数x[i](1 ≤ x[i] ≤ 1000)

输出描述:

输出一个整数,表示牛牛在满足所有奶牛的喜好上编号的方法数。因为答案可能很大,输出方法数对1,000,000,007的模。

输入例子1:

4

4 4 4 4

输出例子1:

24

12.[编程题] 奇异数

时间限制：1秒

空间限制：32768K

如果一个数字满足以下条件,我们就称它为奇异数:

1、这个数字至少有两位

2、这个数的最低两位是相同的

比如: 1488是一个奇异数,而3、112不是。

牛牛现在给出一个一个区间[L,R],让你计算出在区间内有多少个奇异数

输入描述:

输入包括两个正整数L和R(1 ≤ L ≤ R ≤ 10^12),以空格分割。

输出描述:

输出一个正整数,即区间内的奇异数的个数。

输入例子1:

10 20

输出例子1:

1

13.[单选题]

设置tcp的哪个socket参数会影响了 nagle算法？

TCP\_MAXSEG

TCP\_KEEPALIVE

TCP\_SYNCNT

TCP\_NODELAY

答案：

1.A

2.D

3.C

4.C

5.B

6.B

7.B

8.C

9.D

10.#include<stdio.h>

#include<string.h>

int main()

{

intr = 0, i = 0, n, min = 50, r2 = 0, g2 = 0, x;

chars[51], \*p;

gets(s);

for(i = 0; 'R'== s[i]; i++);//去掉头部的R

p = s + i;

n = strlen(p);

for(i = n - 1; i >= 0 && 'G'== p[i]; i--);//去掉尾部的G

n = i + 1;

//统计r的数量，其实应该放到下面的else里面

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

{

if('R'== p[i])

{

r++;

}

}

if(0 == n)

{

min = 0;

}

else

{

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

{

//统计0 ~ i 的r和g的数量

if('R'== p[i])

{

r2++;

}

else

{

g2++;

}

x = g2 + r - r2;//以i为界限，左涂g右涂r的数量

min > x && (min = x);

}

}

//考虑全涂成一色的情况

if(r < min)

{

min = r;

}

if(n - r < min)

{

min = n - r;

}

printf("%d", min);

return0;

}

11.#include <bits/stdc++.h>

using namespace std;

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

{ int cow\_nums;

cin >> cow\_nums;

int x[cow\_nums];

for (int i = 0; i < cow\_nums; ++i) {

cin >> x[i];

}

sort(x, x + cow\_nums);

long long res = 1;

for (int i = 0; i < cow\_nums; ++i) {

res \*= (x[i] - i);

res %= 1000000007;

}

cout << res;

}

12.//利用 0 ~ R 之间的数量与 0 ~ L-1 之间的数量做差

/\*

0 ~ n之间的数量比较容易算，

n/100\*10 ： 比如说1314，就是1300之前的奇异数的个数，

n%100/11+1 : 就是上面的14以内的奇异数的个数，

这样算的话其实0也算，不过在做减法的时候消掉了

\*/

#include<stdio.h>

int main()

{

long L, R;

scanf("%ld%ld", &L, &R);

printf("%ld", (R / 100 - L / 100) \* 10 + R % 100 / 11 - (L - 1) % 100 / 11);

//前面的L忘记减1，可能没测试那个的用例所以AC了.(比如[100,100])

return 0;

}

13.D