２０２１年ＣＳＰ－J 真题题解

单项选择题(共 15 题，每题 2 分，共计 30 分;每题有且仅有一个正确选项)

1.以下不属于面向对象程序设计语言的是()。

A.C++　　B. Python　　C.Java　　D.C

正确答案： D

解析：C 语言是面向过程的语言

2.以下奖项与计算机领域最相关的是()。

A.奥斯卡奖　　B.图灵奖

C.诺贝尔奖　　D.普利策奖

正确答案： B

解析：为表彰英国计算科学家对计算机发展的巨大贡献，旨在奖励对计算机事业作出重要贡献的个人

3.目前主流的计算机储存数据最终都是转换成()数据进行储存。

A.二进制　　B.十进制

C.八进制　　D.十六进制

正确答案：A

解析：计算机的基本信号就是 0 跟 1 。

4.以比较作为基本运算，在 N 个数中找出最大数，最坏情况下所需要的最少的比较次数为

A. N^2　　B.N　　C.N-1　　D. N+1

正确答案： C

解析:让第一个数作为基准值，和后面 N-1 个数分别进行 N-1 次比较。

5.对于入栈顺序为 a，b，c，d，e 的序列，下列()不是合法的出序列。

A.a, b,c,d,e　　B.e,d,c,b,a

C. b, a, c,d, e　　D.c,d,a,e, b

正确答案： D

解析:D 选项中，c 和 d 如果先出栈，那么 a 和 b 要在栈里等候，因为 a 先入栈，b 后入栈，因此 b 肯定要先于 a 出栈。因此 D 不对

6.对于有 n 个顶点、m 条边的无向连通图(m>n)，需要删掉( )条边才能使其成为一棵树。

A.n-1　　B.m-n　　C.m-n-1　　D.m-n+1

正确答案： D

解析:要变成一棵树，最后会保留 n-1 条边。共 m 条边，所以要删去 m-(n-1)=m-n+1 条边。

7.二进制数 101.11 对应的十进制数是()。

A. 6.5　　B. 5.5　　C. 5.75　　D. 5.25

正确答案： C

解析:小数部分 0.1 对应十进制的 1/2，0.01 对应十进制的 1/4。因此小数部分为 0.5+0.25=0.75，因此选 C

8.如果一棵二叉树只有根结点，那么这棵二叉树高度为 1。请问高度为 5 的完全二叉树有( )种不同的形态?

A.16　　B. 15　　C.17　　D. 32

正确答案： A

解析:第 1 层有 1 个结点，第 2 层有 2 个结点，第 3 层有 4 个结点，...，第 5 层有 24-16 个结点。完全二叉树最后一层的结点一定要从左向右排列，不能间断，因此最少有 1 个结点，最多有 16 个结点。(最多是满二叉树，而满二叉树是完全二叉树的一个特殊情况。)

9.表达式 a*(b+c)*d 的后缀表达式为()，其中“*”和“+”是运算符。

A.**a+bcd　　B.abc+*d*

C.abc+d**　　D.*a*+bcd

正确答案：B

解析：遇到字母进栈，遇到符号，就把顶的两个取出来，进行相应的运算，再重新放回栈里。注意运算时是第 2 出栈的+或*第 1 出栈的。

10.6 个人，两个人组一队，总共组成三队，不区分队伍的编号。不同的组队情况有()

种。

A. 10　　B.15　　C.30　　D.20

正确答案：B

解析：解若考虑队伍间先后顺序，第 1 个队伍，有 C 种选法;第 2 个队伍，有 C 种选法;剩下第 3 个队伍，只有 1 种选法。因此共 $C^2_6*C^2_4*1=90$ 种。但是因为不区分队伍的编号，所以每一种队伍会出现 3*2*1 次，比如 12,34,56|12,56,34 34,12,5634,56,12|56,12,34|56,34,12。因此答案为 90/(3*2*1)=15 种 。

11.在数据压缩编码中的哈夫曼编码方法，在本质上是一种( )的策略

A.枚举　　B.贪心　　C.递归　　D.动态规划

正确答案：B

解析：哈夫曼树本质上是出现频率多的点进入到二叉树深度低的叶子节点，出现频率少的点进入到深度大的叶子节点。

12.由 1，1，2，2，3 这五个数字组成不同的三位数有( )种。

A. 18　　B. 15　　C. 12　　D.24

正确答案：A

解析:暴力枚举。可以优化一下枚举方式不重复数字的:3*2*1 种重复数字的:(1)两个 1，一个 2(2)两个 1，一个 3(4)两个 2，一个 3 。共 4 种情况(3)两个 2，-个 1 每种有 3 个可能的三位数。因此答案是 3*2*1+4*3=18

13.考虑如下递归算法

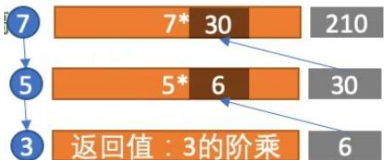
```
solve(n)
if n<=1 return 1
else if n>=5 return n*solve(n-2)
else return n*solve(n-1)
```

则调用 solve(7)得到的返回结果为()。

A.105　　B.840　　C.210　　D.420

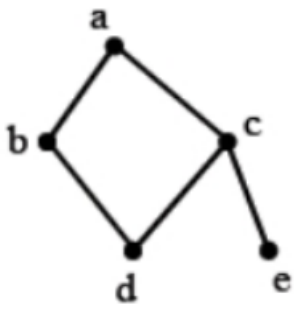
正确答案：A

解析:画分支图草稿



14.以 a 为起点，对右边的无向图进行深度优先遍历，则 b、c、d、e 四个点中有可能作为最后一个遍历到的点的个数为()。

A. 1　　B.2　　C.3　　D.4



正确答案：B

解析:注意是必须从 a 出发。只有 b 和 e 可以作为最后一个遍历到的点

15.有四个人要从 A 点坐一条船过河到 B 点，船一开始在 A 点。该船一次最多可坐两个人。已知这四个人中每个人独自坐船的过河时间分别为 1，2，4，8，且两个人坐船的过河时间为两人独自过河时间的较大者。则最短()时间可以让四个人都过河到 B 点(包括从 B 点把船开回 A 点的时间)

A.14　　B.15　　C.16　　D.17

正确答案：B

解析:信竞经典问题:过河问题，初赛真题原题是用搜索写的。

原则:贪心。

1) 先让 1 和 2 过河到 B，然后 1 把船开回 A 点

2) 然后 4 和 8 过河到 B，然后 2 把船开回 A 点

3) 让 1 和 2 过河到 B，完毕。

4) 答案为(2+1)+(8+2)+2=15.关键点在于 2)，要让大的吸收大的，即取 max(4,8)否则就会出现 4+8。

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 x;除特殊说明外，判断题 1.5 分，选择题 3 分，共计 40 分)

(1)

```
01 #include <iostream>
02 using namespace std;
03
04 int n;
05 int a[1000];
06
07 int f(int x)
08 {
09     int ret = 0;
10     for (; x; x &= x - 1) ret++;
11     return ret;
12 }
13
14 int g(int x)
15 {
16     return x & -x;
17 }
18
19 int main()
20 {
21     cin >> n;
22     for (int i = 0; i < n; i++) cin >> a[i];
23     for (int i = 0; i < n; i++)
24         cout << f(a[i]) + g(a[i]) << ' ';
25     cout << endl;
26     return 0;
27 }
```

判断题

16.输入的 n 等于 1001 时，程序不会发生下标越界。()

17.输入的 a[i]必须全为正整数，否则程序将陷入死循环。

18.当输入为“5 211 916 10”时，输出为“3 4 3 17 5”()

19.当输入为“1 511998”时，输出为“18”()

20.将源代码中 g 函数的定义(14-17 行)移到 main 函数的后面，程序可以正常编译运行。()

单选题

21.当输入为“2-65536 2147483647”时，输出为()。

A.“65532 33” B.“65552 32"

C.“65535 34" D.“65554 33"

解析:重点在于 f 和 g 函数分别是什么意思。可以把第 18 题的输入“52119 16 10”拿来尝试一下。

x (10进制)		2	11	9	16	10
第1次循环	x	**10**	**1011**	**1001**	**10000**	**1010**
	x-1	01	1010	1000	01111	1001
	x & (x-1)	00	1010	1000	00000	1000
第2次循环	x		1010	1000		1000
	x-1		1001	0111		0111
	x & (x-1)		1000	0000		0000
第3次循环	x		1000			
	x-1		0111			
	x & (x-1)		0000			
循环次数		**1**	**3**	**2**	**1**	**2**

比对循环次数和 x 的二进制的关系，可以发现 f 函数实际上是求:x 的二进制里 1 的个数。

g 函数:-x 是把 x 的二进制按位取反后+1。依然可以把第 18 题的输入“52 119 16 10”拿来尝试一下。

x (10进制)	2	11	9	16	10
x	**10**	**1011**	**1001**	**10000**	**1010**
把x补成8位	00000010	00001011	00001001	00010000	00001010
按位取反	11111101	11110100	11110110	11101111	11110101
再+1(即-x)	11111110	11110101	11110111	11110000	11110110
x&(-x)	00000010	00000001	00000001	00010000	00000010
g(x)转为10进制	2	1	1	16	2
f(x)	1	3	2	1	2
f(x)+g(x)	3	4	3	17	4

可以发现 g(x)的结果为:只保留 x 的二进制里的最后一个 1，而形成的新的二进制数字。

判断题解析：

16.输入的 n 等于 1001 时，程序不会发生下标越界。()

正确答案：正确

解析:数组下标只能到 999，因此会越界

17.输入的 a[i]必须全为正整数，否则程序将陷入死循环。

正确答案：错误

解析:负数也可以位运算。

18.当输入为“5 211 916 10”时，输出为“3 4 3 17 5”()

正确答案：错误

解析:根据以上草稿和分析，答案为 3 4 3 17 4.

19.当输入为“1 511998”时，输出为“18”()

正确答案：正确

解析:对 511998 进行二进制分解得

1111100111111111110.

根据以上分析，f(x)指的是二进制里有多少个 1，因此 f(x)=g(x)

指的是只保留最后一个 1，即 00000000000000010，即二进制的 2.因此答案为 16+2=18。

20.将源代码中 g 函数的定义(14-17 行)移到 main 函数的后面，程序可以正常编译运行。()

正确答案：错误

解析:要移到后面，需要提前声明函数。

21.当输入为“2-65536 2147483647”时，输出为()。

A.“65532 33”　　B.“65552 32"

C.“65535 34"　　D.“65554 33"

正确答案：B

解析:第 2 个变量 2147483647 为 int 的最大值比较好想象，因此从这里入手。2147483647 写成 2 进制，是 01111111111111111111111111111111，1 个 0+31 个 1(第 1 个 0 是符号位)。根据以上分析，f(x)是 31，g(x)是 1，因此答案为 32，因此选 B。

```cpp
#include <iostream>
#include <string>
using namespace std;

char base[64];
char table[256];

void init()
{
    for (int i = 0; i < 26; i++) base[i] = 'A' + i;
    for (int i = 0; i < 26; i++) base[26 + i] = 'a' + i;
    for (int i = 0; i < 10; i++) base[52 + i] = '0' + i;
    base[62] = '+', base[63] = '/';

    for (int i = 0; i < 256; i++) table[i] = 0xff;
    for (int i = 0; i < 64; i++) table[base[i]] = i;
    table['='] = 0;
}

string decode(string str)
{
    string ret;
    int i;
    for (i = 0; i < str.size(); i += 4) {
        ret += table[str[i]] << 2 | table[str[i + 1]] >> 4;
        if (str[i + 2] != '=')
            ret += (table[str[i + 1]] & 0x0f) << 4 | table[str[i +
                                                               2]] >> 2;
        if (str[i + 3] != '=')
            ret += table[str[i + 2]] << 6 | table[str[i + 3]];
    }
    return ret;
}

int main()
{
    init();
    cout << int(table[0]) << endl;

    string str;
    cin >> str;
    cout << decode(str) << endl;
    return 0;
}
```


1. 判断题

输出的第二行一定是由小写字母、大写字母、数字和 +、/、= 构成的字符串。（ ）

可能存在输入不同，但输出的第二行相同的情形。（ ）

输出的第一行为 -1。（ ）

2. 单选题

1. 设输入字符串长度为 n，`decode` 函数的时间复杂度为（ ）

A. $O(n)$ B. $O(\text{sqrt}(n))$ C. $O(n\log n)$ D. $O(n^2)$

2. 当输入为 Y3Nx 时，输出的第二行为（）。（3.5 分）

A. `csp` B. `csq` C. `CSP` D. `Csp`

3.当输入为 Y2NmIDIwMjE= 时，输出的第二行为（ ）。

A. `ccf2021`

B. `ccf2022`

C. `ccf 2021`

D. `ccf 2022`

解析：我们观察一下 init()函数

```
36    init();
37    cout << int(table[0]) << endl;
```

然后我们输入了一个 str 字符串，通过我们的 decode 函数获取新的字符串 ret，我们输出这个字符串

```
39      string str;
40      cin >> str;
41      cout << decode(str) << endl;
```

那么我们 init 函数做什么事情呢？

```
08 void init()
09 {
10     for (int i = 0; i < 26; i++) base[i] = 'A' + i;
11     for (int i = 0; i < 26; i++) base[26 + i] = 'a' + i;
12     for (int i = 0; i < 10; i++) base[52 + i] = '0' + i;
13     base[62] = '+', base[63] = '/';
14
15     for (int i = 0; i < 256; i++) table[i] = 0xff;
16     for (int i = 0; i < 64; i++) table[base[i]] = i;
17     table['='] = 0;
18 }
```

我们发现 base 数组的显示是

0	1	2	3	4	5	6	7	8	9	10	11	12
A	B	C	D	E	F	G	H	I	J	K	L	M
13	14	15	16	17	18	19	20	21	22	23	24	25
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
26	27	28	29	30	31	32	33	34	35	36	37	38
a	b	c	d	e	f	g	h	i	j	k	l	m
39	40	41	42	43	44	45	46	47	48	49	50	51
n	o	p	q	r	s	t	u	v	w	x	y	z
52	53	54	55	56	57	58	59	60	61	62	63	
0	1	2	3	4	5	6	7	8	9	+	=	

我们可以基于此确定数字跟字母的 base 码(是个数);

当然这个是输入数，输出对应 base 码的 char 字符

我们还需要输入字符，输出的是 base 码(base 码是我编的)

Table 就是这么干的

```
15      for (int i = 0; i < 256; i++) table[i] = 0xff;
16      for (int i = 0; i < 64; i++) table[base[i]] = i;
17      table['='] = 0;
```

但是 BASE 码的词典里只有 63 个啊？Table 数组里剩下的东西等于多少呢？

诶对，等于 0xff char 类型的 0xff 等于多少呢？对等于-1;所以不在 base 里词典的那就是-1。

我们接下来看看 decode(str)

```
20 string decode(string str)
21 {
22      string ret;
23      int i;
24      for (i = 0; i < str.size(); i += 4) {
25          ret += table[str[i]] << 2 | table[str[i + 1]] >> 4;
26          if (str[i + 2] != '=')
27              ret += (table[str[i + 1]] & 0x0f) << 4 | table[str[i +
                                                              2]] >> 2;
28          if (str[i + 3] != '=')
29              ret += table[str[i + 2]] << 6 | table[str[i + 3]];
30      }
31      return ret;
32 }
```

我们发现字符串 str 是每四个字符输出一组 ret

每组 ret 根据是不是等号输出 1~3 个字符，这些字符本质上是 **table[str[i]]**的数组拼接而成!!

我们发现每次平移都是 2 的倍数，所以我们可以两个 bit 位简化为一个 bit 位!!

我们发现 table 数组输出的数值<=63.所以

table[str[i]]	0	X12	X13	X14
table[str[i+1]]	0	X22	X23	X24
table[str[i+2]]	0	X32	X33	X34
table[str[i+3]]	0	X42	X43	X44

最高的一个位就是 0.

我们来看看循环里的操作

```
25        ret += table[str[i]] << 2 | table[str[i + 1]] >> 4;
```

25 行的 ret+=

X12	X13	X14	0
0	0	0	X22

我们发现 ret+ 等于

X12	X13	X14	X22

我们来看一下一行

```
26        if (str[i + 2] != '=')
27            ret += (table[str[i + 1]] & 0x0f) << 4 | table[str[i +
                                                              2]] >> 2;
```

table[str[i+1]&0x0f]<<4

我们知道与运算符相当于*1，任何数与 1 得任何数，

任何数与 0 得零所以这个结果就是

X23	X24	0	0
0	0	X32	X33

我们发现 ret+ 等于

X23	X24	X32	X33

OK!!到了我们的最后一句了

```
28          if (str[i + 3] != '=')
29              ret += table[str[i + 2]] << 6 | table[str[i + 3]];
```

X34	0	0	0
0	X42	X43	X44

我们发现 ret+ 等于

X34	X42	X43	X44

OK!!我们汇个总

Ret 序	条件	1	2	3	4
1	无	X12	X13	X14	X22
2	str[i+2]!= ‘=’	X23	X24	X32	X33
3	str[i+3]!=’=’	X34	X42	X43	X44

1.判断题

输出的第二行一定是由小写字母、大写字母、数字和 +、/、= 构成的字符串。（ ）

正确答案：错误

解析：输出的是 ret，ret 的范围是 char 型数组，且不保证值域收敛，有可能是 char 型所有的符号满天飞

可能存在输入不同，但输出的第二行相同的情形。（ ）

正确答案：正确

解析：我们假设 str[i] 不属于大小写字母数字，等号，加号，那么无论 str[i]等于什么，table[str[i]] 永远等于-1;

输出的第一行为 -1。（ ）

正确答案：正确

解析： table[0]，base 码里没有 0 的选项，所以答案是 table[0] = 0xff,

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

由于是 char 型变量，所以符号位为 1,这是个负数，负数输出它的补码等于原码取反+1;

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	反
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	补

所以输出是-1。

2.选择题

1. 设输入字符串长度为 n，decode 函数的时间复杂度为（ ）

A. $O(n)$ B. $O(\sqrt{n})$ C. $O(n\log n)$ D. $O(n^2)$

正确答案：A

解析：

```
20 string decode(string str)
21 {
22     string ret;
23     int i;
24     for (i = 0; i < str.size(); i += 4) {
25         ret += table[str[i]] << 2 | table[str[i + 1]] >> 4;
26         if (str[i + 2] != '=')
27             ret += (table[str[i + 1]] & 0x0f) << 4 | table[str[i +
                                                              2]] >> 2;
28         if (str[i + 3] != '=')
29             ret += table[str[i + 2]] << 6 | table[str[i + 3]];
30     }
31     return ret;
32 }
```

每一轮输入 4 个字符 ret 输出 1~3 个字符，所以正确答案选 A

2. 当输入为 Y3Nx|时，输出的第二行为（）。（3.5 分）

A. csp B. csqC. CSP D. Csp

正确答案：B

解析：我们对照我们打的 table 码表(base 的反表)

0	1	2	3	4	5	6	7	8	9	10	11	12
A	B	C	D	E	F	G	H	I	J	K	L	M
13	14	15	16	17	18	19	20	21	22	23	24	25
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
26	27	28	29	30	31	32	33	34	35	36	37	38
a	b	c	d	e	f	g	h	i	j	k	l	m
39	40	41	42	43	44	45	46	47	48	49	50	51
n	o	p	q	r	s	t	u	v	w	x	y	z
52	53	54	55	56	57	58	59	60	61	62	63	
0	1	2	3	4	5	6	7	8	9	+	=	

table['Y'] = 24 table['3'] =55 table['N'] = 13 table ['x'] = 49;

所以我们的 table[str[0~3]]是:

table[str[0]]	0	1	2	0
table[str[1]]	0	3	1	3
table[str[2]]	0	0	3	1
table[str[3]]	0	3	0	1

那么我们的 ret+就是

ret 序	4	3	2	1
1	1	2	0	3
2	1	3	0	3
3	1	3	0	1

这是我们四进制的表示方式

那么

ret1 = 1*4^3+2*4^2+0*4+3 = 99 ,'c'的 ASCII 码是 99;

ret2 = 1*4^3+3*4^2+0*4+3 = 115,'s'的 ASCII 码是 115

ret3 = 1*4^3+3*16+0*4+1= 113,'q'的 ASCII 码是 113;

3.当输入为 Y2NmIDIwMjE=时，输出的第二行为（ ）。

A. ccf2021

B. ccf2022

C. ccf 2021

D. ccf 2022

正确答案：C

解析：我们每四个分一组

ret 贡献值	Y2Nm	IDIw	MjE=
输出的字符	3	3	2

我们发现 ret 输出是 8 个字符，所以 AB 错误

我们发现答案最后一项有差异！！

Table['j'] = 35 ==(203)4

table['E'] =4 ==(010)4

最后一位的 ret

X23	X24	X32	X33

等于

0	3	0	1

所以答案是 3*16+1 = 49;ASCII['1'] = 49;最后一项是 1，

D 选项错误

综上答案是 C

(3)

```
01 #include <iostream>
02 using namespace std;
03
04 const int n = 100000;
05 const int N = n + 1;
06
07 int m;
08 int a[N], b[N], c[N], d[N];
09 int f[N], g[N];
10
11 void init()
12 {
13     f[1] = g[1] = 1;
14     for (int i = 2; i <= n; i++) {
15         if (!a[i]) {
16             b[m++] = i;
17             c[i] = 1, f[i] = 2;
18             d[i] = 1, g[i] = i + 1;
19         }
20         for (int j = 0; j < m && b[j] * i <= n; j++) {
21             int k = b[j];
22             a[i * k] = 1;
23             if (i % k == 0) {
24                 c[i * k] = c[i] + 1;
25                 f[i * k] = f[i] / c[i * k] * (c[i * k] + 1);
26                 d[i * k] = d[i];
```



```
27                 g[i * k] = g[i] * k + d[i];
28                 break;
29             }
30             else {
31                 c[i * k] = 1;
32                 f[i * k] = 2 * f[i];
33                 d[i * k] = g[i];
34                 g[i * k] = g[i] * (k + 1);
35             }
36         }
37     }
38 }
```

```
40 int main()
41 {
42     init();
43
44     int x;
45     cin >> x;
46     cout << f[x] << ' ' << g[x] << endl;
47     return 0;
48 }
```

假设输入的 x 是不超过 1000 的自然数，完成下面的判断题和单选题:

判断题

28.若输入不为“1”，把第 13 行删去不会影响输出的结果。()

29.(2 分)第 25 行的“f[i]/ c[i *k]”可能存在无法整除而向下取整的情况。()

30.(2 分)在执行完 init()后，f 数组不是单调递增的，但 g 数组是单调递增的。()

单选题

31. init 函数的时间复杂度为()。

A. O(n)　　B.e(nlogn)

C.O(n √ n)　　D. $O(n^2)$

32.在执行完 init()后，f[1]，f[2]，f[3]… f[100]中有( )个等于 2

A.23　　B.24　　C.25　　D.26

33.(4 分)当输入为“1000”时，输出为()。

“15 1340”B.“15 2340”“16 2340”D.“16 1340"

程序解析：本题是经典算法欧拉筛，比常用筛法埃氏筛法难理解几倍。思维难度高，长时间不接触一定会忘，对于入门级来说其实比较苛刻。而且本题不是模板，还要在此基础上进行深度分析，

```
11 void init()
12 {
13     f[1] = g[1] = 1;
14     for (int i = 2; i <= n; i++) {// 枚举倍数i,
15         if (!a[i]) {                  // 也兼具枚举质数合数
16             b[m++] = i; // b数组存放质数列表
17             c[i] = 1, f[i] = 2;
18             d[i] = 1, g[i] = i + 1;
19         }                          // 质数*倍数<=n，标记
20         for (int j = 0; j < m && b[j] * i <= n; j++) {
21             int k = b[j]; // k：当前质数
22             a[i * k] = 1; // 标记质数k的i倍为合数
23             if (i % k == 0) { // 遇到i的最小质因数,
24                 c[i * k] = c[i] + 1; 就停止枚举质数列表
25                 f[i * k] = f[i] / c[i * k] * (c[i * k] + 1);
26                 d[i * k] = d[i];
27                 g[i * k] = g[i] * k + d[i];
28                 break;
29             }
```

c 是什么意思呢?24 句的 k 是 i*k 的最小质因数,每次+1，说明 c[i*k]记录最小质因数的个数。例如:48=24*3 中 2 是 48 的最小质因数，有 4 个，因此 c[48]=4。第 31 句时，因为还没到 i 的最小质因数，因此 k 一定是 i*k 的第 1 个最小质因数。

f 是什么意思呢？例如:k=2，i=24 倍。i*k=48。f[48]=f[24]/c[48]*(c[48]+1)。因为 48=24*3，因此 c[48]=4，即 f[48]=f[24]/4*5。24 和 48 只相差一个因数 2，那么在因数个上 24=23*3，即(3+1)*(1+1)个因数，而

48=24*3，即(4+1)*(1+1)个因数,

因此，48 的因数个数/(4+1)=24 的因数个数/(3+1)。

d 是什么意思呢?第 33 句的 d[i*k]是遇到 i*k 的最小质因数 k 时，此时的倍数 i 的所有因数之和。第 34 句 g[i*k]是 i*k 的所有因数的和。例如 g[9*2]=g[9]*(2+1)。为什么呢?9 的因数有 3^0,3^1,3^2，9*2 的因数有 3^0,3^1,3^2,$2*3^0$,$2*3^1$,$2*3^2$，加起来是是(2+1)(30+31+32)，即(k+1)*g[i]。

a 标记是不是质数，b 是质数的列表。c 表示最小质因数的个数，f 是因数个数，g 是因数和，

这些数组的是什么意思出来之后，就可以做题目了

1.判断题解析

28.若输入不为“1”，把第 13 行删去不会影响输出的结果。()

正确答案：正确

解析:程序后面没有要用到 f[1]或 g[1]的地方，因此不影响。

29.(2 分)第 25 行的“f[i]/ c[i *k]”可能存在无法整除而向下取整的情况。()

正确答案：错误

解析:根据以上分析，c[i*k]表示 i*k 的最小质因数(即 k)的个数，而 f[表示 i 的因数的个数，举例 48=24*3，因此 c[48]=4。现在 k=2，i=24 倍，24 的因数的个数是否一定能整除 4 呢?答案是肯定的，因此 $24=2^3*3$，24 的因数里 2 的可能情况是 3+1 种(2^0,2^1,2^2,2^3)，因此 f[24]确实能整除 c[28]。

30.(2 分)在执行完 init()后，f 数组不是单调递增的，但 g 数组是单调递增的。()

正确答案：错误

解析:根据以上分析，f 数组表示因数个数，g 数组表示因数之和。这两个都不是单调的。

2.选择题解析

31. init 函数的时间复杂度为()。

A. O(n)　　B.e(nlogn)

C.O(n √ n)　　D. $O(n^2)$

正确答案：A

解析:欧拉筛即线性筛，因为每个合数仅会被标记一次，因此时间复杂度是 O(N)

32.在执行完 init()后，f[1]，f[2]，f[3]… f[100]中有( )个等于 2

A.23　　B.24　　C.25　　D.26

正确答案：C

解析:f[]的含义是 i 有多少个因数。因此本题就是说因数是 2 个的有多少个，而只有质数有 2 个因数，即找到 100 以内的质数的个数。100 以内的质数有 25 个

33.(4 分)当输入为“1000”时，输出为()。

“15 1340” B.“15 2340”“16 2340” D.“16 1340"

正确答案：C

解析:找 1000 的因数个数和因数之和。

三、完善程序(单选题，每小题 3 分，共计 30 分)

(Josephus 问题)有几个人围成一个圈，依次标号 0 至 n-1。从 0 号开(1)始，依次 0,1,0,1,… 交替报数，报到 1 的人会离开，直至圈中只剩下一个人。求最后剩下人的编号。

```
01 #include <iostream>
02
03 using namespace std;
04
05 const int MAXN = 1000000;
06 int F[MAXN];
07
08 int main() {
09     int n;
10     cin >> n;
11     int i = 0, p = 0, c = 0;
12     while (①) {
13         if (F[i] == 0) {
14             if (②) {
15                 F[i] = 1;
16                 ③;
17             }
18             ④;
19         }
20         ⑤;
21     }
22     int ans = -1;
23     for (i = 0; i < n; i++)
24         if (F[i] == 0)
25             ans = i;
26     cout << ans << endl;
27     return 0;
28 }
```

34.①处应填()

A. i<n　B.C<n　C.i<n-1　D.c<n-1

35.②处应填()

A.i%2==0　B.i%2==1　C.p　D. !p

36.③处应填()

A. i++　B.i=(i+1)%n　C.C++　D.p^=1

37.④处应填()

A. i++　　B.i=(i+1)%n　C.C++ D. p^=1

38.⑤处应填()

A. i++　　B.i=(i+1)%n　　C.c++　　D.p^=1

程序解析

本题是经典程序约瑟夫问题的变形可能同学们拿到题以后，感觉自己做过，会很开心但是实则此题时一个变形问题，要想正确分析

根据 13，15 行，猜测 F[表示 i 有没有报过数，F[i]=0 应该代表没有报过。第 15 行是报到 1 而被标记离开，因此③表示计数 c++。③选 C

为了模拟在一个环上逐一检测这个动作，可以猜一定会有一个找下一个 i 的地方，这个地方应该跟标没标过没有关系，因此这个位置应该是⑤，在环上找下一个肯定要取模，因此⑤选 B。

本题是 01 交替报数，因此猜测②是判断现在报 0 还是报 1 如果报 1 就会停止，因此②选 p==1,即 C。

剩下的④则必须进行 01 交替的操作，即 0 变 1，1 变 0。可以用经典异或运算 p^1 实现，因此④选 D。

最后①就是在控制什么时候停止，可以用个数 c 来实现。举小例子，n=2，那么当检测到 c=1 就应该停止循环，即 c<1 才循环，即 c<n-1，因此①选 D。

34.①处应填(D)

A. i<n　B.C<n　C.i<n-1　D.c<n-1

35.②处应填(C)

A.i%2==0　B.i%2==1　C.p　D. !p

36.③处应填(C)

A. i++　B.i=(i+1)%n　C.C++　D.p^=1

37.④处应填(D)

A. i++　B.i=(i+1)%n　C.C++　D. p^=1

38.⑤处应填(B)

A. i++　B.i=(i+1)%n　C.c++　D.p^=1

(2) (矩形计数)平面上有 n 个关键点，求有多少个四条边都和 x 轴或者 y 轴平行的矩 形，满足四个顶点都是关键点。给出的关键点可能有重复，但完全重合的矩形只计一次。

```
01 #include <iostream>
02
03 using namespace std;
04
05 struct point {
06     int x, y, id;
07 };
08
09 bool equals(point a, point b) {
10     return a.x == b.x && a.y == b.y;
11 }
12
13 bool cmp(point a, point b) {
14     return ①;
15 }
```

```
12
13 bool cmp(point a, point b) {
14     return ①;
15 }
16
17 void sort(point A[], int n) {
18     for (int i = 0; i < n; i++)
19         for (int j = 1; j < n; j++)
20             if (cmp(A[j], A[j - 1])) {
21                 point t = A[j];
22                 A[j] = A[j - 1];
23                 A[j - 1] = t;
24             }
25 }
26
27 int unique(point A[], int n) {
28     int t = 0;
29     for (int i = 0; i < n; i++)
30         if (②)
31             A[t++] = A[i];
32     return t;
33 }
34
35 bool binary_search(point A[], int n, int x, int y) {
36     point p;
37     p.x = x;
38     p.y = y;
39     p.id = n;
40     int a = 0, b = n - 1;
41     while (a < b) {
42         int mid = ③;
43         if (④)
44             a = mid + 1;
45         else
46             b = mid;
47     }
48     return equals(A[a], p);
49 }
50
51 const int MAXN = 1000;
52 point A[MAXN];
53
54 int main() {
55     int n;
56     cin >> n;
57     for (int i = 0; i < n; i++) {
58         cin >> A[i].x >> A[i].y;
59         A[i].id = i;
60     }
61     sort(A, n);
62     n = unique(A, n);
63     int ans = 0;
64     for (int i = 0; i < n; i++)
65         for (int j = 0; j < n; j++)
66             if (⑤ && binary_search(A, n, A[i].x, A[j].y) &&
                    binary_search(A, n, A[j].x, A[i].y)) {
67                 ans++;
```

```
68            }
69    cout << ans << endl;
```



```
70    return 0;
71 }
```

39.①处应填()

A. a.x != b.x ? a.x< b.x : a.id< b.id

B.a.x != b.x ? a.x< b.x : a.y < b.y

C.equals(a,b)?a.id<b.id :a.x< b.x

D.equals(a, b)? a.id < b.id :(a.x != b.x ? a.x< b.x : a.y< b.y)

40.②处应填( )

A.i==0||cmp(A[i],A[i-1])

B.t==0||equals(A[i],A[t -1])

C.i==0||!cmp(A[i],A[i - 1])

D.t == 0|| !equals(A[i],A[t - 1])

41.③处应填()

A. b-(b-a)/2+1　　B.(a+b+1)>>1

C. (a+b)>>1　　D.a+(b-a+1)/2

42.④处应填()

A.!cmp(A[mid],p)　　B.cmp(A[mid],p)

C.cmp(p,A[mid])　　D. !cmp(p,A[mid])

43.⑤处应填( )

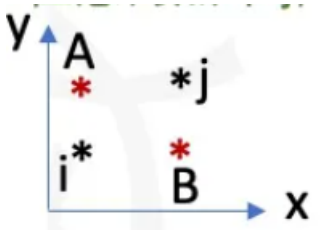
A. A[i].x == Aj].x

B. A[i].id < A[j].id

C. A[i].x ==A[j].x && A[i].id < A[j].id

D. A[i].x< A[j].x && A[i].y< A[j].y

题目解析:通读程序后发现题目在有二分 binary_search,因此是在找什么东西。观察 64，65 行，推断这里的循环是在枚举任意两点 i 和 j，画一下草图



第 66 句说要二分找 i 的 x 和 j 的 y 以及 i 的 y 和 j 的 x。在草图上标记一下会发现，i 的 x 和 j 的 y 组成了 A 点;而 i 的 y 和 j 的 x 组成了 B 点。思路一下就跃然纸上本程序大体思路就是枚举任意两点作为矩形对角线上的两点再寻找另外一条对角线上的两点存不存在

39.①处应填()

A. a.x != b.x ? a.x< b.x : a.id< b.id

B.a.x != b.x ? a.x< b.x : a.y < b.y

C.equals(a,b)?a.id<b.id :a.x< b.x

D.equals(a, b)? a.id < b.id :(a.x != b.x ? a.x< b.x : a.y< b.y)

正确答案：B

解析：观察 sort，发现有交换操作，是冒泡排序，按照 x 和 y 双关键字排序，因此①选 B

40.②处应填( )

A.i==0||cmp(A[i],A[i-1])

B.t==0||equals(A[i],A[t -1])

C.i==0||!cmp(A[i],A[i - 1])

D.t == 0|| !equals(A[i],A[t - 1])

正确答案：D

解析：根据题意，点可能有重复的，unige 函数应该在去重。第 i 个和上一个不一样就加入列表，因此②选 D。

41.③处应填()

A. b-(b-a)/2+1　　B.(a+b+1)>>1

C. (a+b)>>1　　D.a+(b-a+1)/2

正确答案：C

解析：③是找[l~r]的中点，选 C。

42.④处应填()

A.!cmp(A[mid],p)　　B.cmp(A[mid],p)

C.cmp(p,A[mid])　　D. !cmp(p,A[mid])

正确答案：B

解析：④处是说如果怎么样，就向右查找(向大的方向)，那么肯定是 mid 和要找的(x,y)相比小了，因此选 B。

43.⑤处应填( )

A. A[i].x == Aj].x

B. A[i].id < A[j].id

C. A[i].x ==A[j].x && A[i].id < A[j].id

D. A[i].x< A[j].x && A[i].y< A[j].y

正确答案：D

解析：⑤处，因为枚举时 i 和 j 随机，如果不保证 i 和 j-左一右(0 那么 j 和 i 也会产生同样的矩形，因此选 D。

一、单项选择题（共 15 题，每题 2 分，共计 30 分）

1	2	3	4	5	6	7	8	9	10
D	B	A	C	D	D	C	A	B	B
11	12	13	14	15					
B	A	C	B	B					

二、阅读程序（除特殊说明外，判断题 1.5 分，单选题 3 分，共计 40 分）

	判断题（填√或×）					单选题
第 1 题	16)	17)	18)	19)	20)	21)
	×	×	×	√	×	B
	判断题（填√或×）			单选题		
第 2 题	22)	23)	24)	25)	26)	27) (3.5 分)
	×	√	√	B	B	C
	判断题（填√或×）			单选题		
第 3 题	28)	29) (2 分)	30) (2 分)	31)	32)	33)(4 分)
	√	×	×	A	C	C

三、完善程序（单选题，每小题 3 分，共计 30 分）

第 1 题					第 2 题				
34)	35)	36)	37)	38)	39)	40)	41)	42)	43)
D	C	C	D	B	B	D	C	B	D