2021 CCF 非专业级别软件能力认证第一轮

（CSP-J1）入门级 C++语言试题解析 - hebin

认证时间：2021 年 9 月 19 日 14:30~16:30

考生注意事项：

● 试题纸共有 12 页，答题纸共有 1 页，满分 100 分。请在答题纸上作答，写在试题纸上的一律无效。

● 不得使用任何电子设备（如计算器、手机、电子词典等）或查阅任何书籍资料。

一、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

答案：D，C++、AVA、Python 都是面向对象程序设计语言，C 是面向过程的程序设计语言。

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

答案：B，图灵奖是计算机领域的最高奖项；

奥斯卡奖是电影类奖项，奥斯卡小金人；

诺贝尔奖是指根据诺贝尔 1895 年的遗嘱而设立的五个奖项，包括:物理学奖、化学奖、和平奖、生理学或医学奖和文学奖，旨在表彰在物理学、化学、和平、生理学或医学以及文学上"对人类作出最大贡献"的人士;以及瑞典中央银行 1968 年设立的诺贝尔经济学奖,用于表彰在经济学领域杰出贡献的人。

普利策奖，又名普利策新闻奖，是根据美国报业巨头约瑟夫·普利策的遗愿于 1917 年设立的奖项，后发展成为美国新闻界的最高荣誉奖。

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

答案：A，计算机只能识别二进制，所以全部的数据都是最终转为二进制存储的。

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

答案：C，N 个数中找到某个数的最坏比较次数为 N-1，也就是和其余的数都比较一次。

5. 对于入栈顺序为 a，b，c，d，e 的序列，下列（ ）不是合法的出栈序列。

A. a，b，c，d，e

B. e，d，c，b，a

C. b，a，c，d，e

D. c，d，a，e，b

答案：D，方法一：将选项模拟一遍即可；

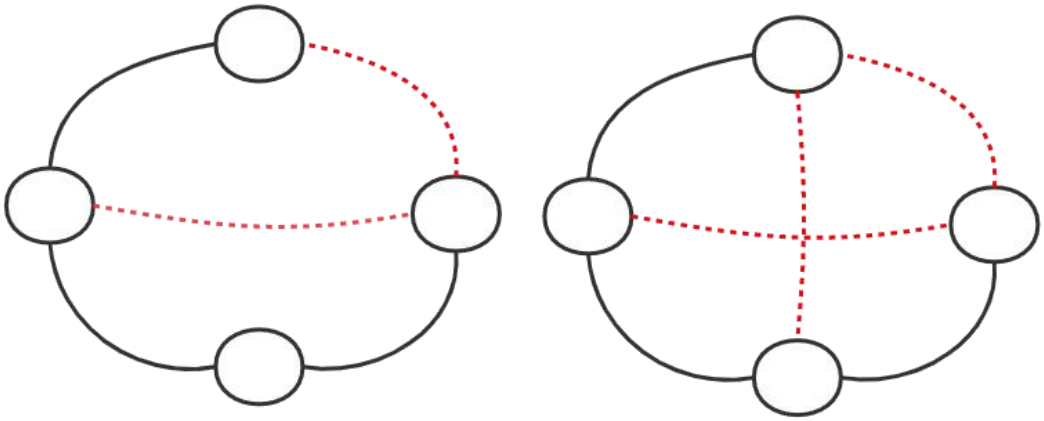
方法二，由于入栈顺序为字典序递增，那么出栈顺序如果出现字典序下降的情况，则从第一位字典序下降的元素开始，应当出现字典序顺序递减，直到栈空。

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

答案：D，简单方法：举例排除

当 n=4, m=5, 删除 2 条边； 当 n=4, m=6，删除 3 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

答案：C，按照数值与权值的乘积累加和计算即可；

101.11B --- 从左至右权值分别为：2^2, 2^1, 2^0, 小数点，2^(-1), 2^(-2)

101.11B = 2^2+2^0+2^(-1)+2^(-2)=5.75

扩展：x 进制的写法：101(x)； 5.75D 如何转为二进制

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

答案：A，当一棵二叉树的根节点高度为 1 时，有这两个公式需要记住：

1. 当高度为 k 时，第 k 层的节点个数最多为 2^{k-1}

2. 当高度为 k 时，这棵树的节点数量最多为 $2^k - 1$

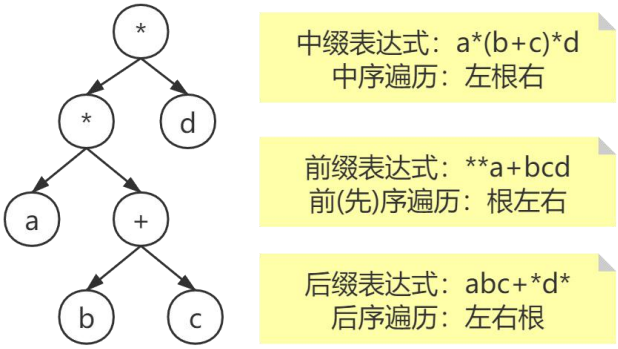
所以本题中，当 k=5，第 k 层的节点数最多为 16，由于完全二叉树的树形是在满二叉树的叶子节点上从右至左进行删除，且最少第 k 层有一个节点，所以一共有 16 中不同形态。

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

答案：B，前缀、中缀、后缀三种表达式分别对应二叉树的先序、中序、后序遍历；

先根据中缀表达式建立二叉树，再按照后序遍历对树进行遍历。



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

答案：B，6 个人组成三队，每队 2 人，那么可以按照组合：$C_6^2\,C_4^2C_2^2 = 90$

但是不区分队伍编号，也就是 3 个队伍之间没有先后，如：123，321 其实是同样的组合数列，而这样的排列情况有：$A_3^3 = 3! = 6$

所以最后结果为：90/6=15。

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

答案：B，哈夫曼编码基于信源的概率统计模型，它的基本思路是，出现概率大的信源符号编短码，出现概率小的信源符号编长码，从而使平均码长最小。

这是一种贪心策略，每次选取当前概率最大的符号使用现有的最短码。

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

答案：A，我没有相到什么好的解法，就是一个一个枚举出来，结果为 18 种，

但是为了防止出现重复和丢失的情况，规定数据升序排列：

112，113，121，122，123，131，132
211，212，213，221，223，231，232
311，312，321，322

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

答案：C，将 7 带入，ans = 7*solve(5)

将 5 带入， ans = 7*5*solve(3)

将 3 带入，ans = 7*5*3*solve(2)

将 2 带入，ans = 7*5*3*2*solve(1)

将 1 带入，ans = 7*5*3*2*1 = 210

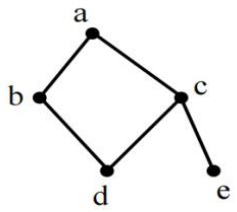
14.以 a 为起点，对右边的无向图进行深度优先遍历，则 b、 c、 d、 e 四个点中有可能作为最后一个遍历到的点的个数为（ ）。

A. 1

B. 2

C. 3

D. 4



答案：2，从 a 开始 dfs,可以（ace）或者（acdb）或者（abdce），所以就只有 be 两个点。

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

答案：B，这个题感觉找不到准确的解法，只能一个一个试啊！

（原谅我这样想，确实想不到有什么直接的解法）。

左边	河	右边	时间
1，2，4，8	-> 1，2	1,2	2
2,4,8	<- 2	1	2
2	-> 4， 8	1,4,8	8
1,2	<- 1	4,8	1
	-> 1，2	1,2,4,8	2

再试一试其他的，没发现更小的了，就选个（2+2+8+1+2=15）,蒙一个吧！

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)

```
1   #include <iostream>
2   using namespace std;
3
4   int n;
5   int a[1000];
6
7   int f(int x)
8   {
9       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 2 11 9 16 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”

答案：16-21：××××√× B

16. n=1001，下标就会取到 1000，但是数组开 int a[1000]，下标只能到 999，所以越界了。

17. 对程序分析发现，f(x)是对 x 的二进制中 1 的个数进行计数，g(x)是一个 lowbit(x)，表示取 x 的二进制中最右边的 1 所对应的权值，所以结果不会陷入死循环。

重点记录：f(x)与 g(x)的写法，在实际应用中或许一时想不到位运算的写法，很少这样写，但是关于位运算的技巧，大家可以了解一些，同时对于 $x\&(-x)$ 作如下说明：

一个数的负数等于这个数的正数取反后+1，所以：$x\&(-x) = x\&(\sim x+1)$。

18. 最简单的方法是带入一个一个计算，当然，程序读懂了就可以直接写结果

5	2	11	9	16	10
x, f(x)	2=10B, 1	11=1011B, 3	9=1001B, 2	16=10000B, 1	10=1010B, 2
g(x)	2	1	1	16	2
ans	3	4	3	17	4

19. x = 511998
= 262144 + 131072 + 65536
+ 32768 + 16384
+ 2048 + 1024 + 512 + 256
+ 128 + 64 + 32 + 16
+ 8 + 4 + 2. 二进制表示 ==》

所以 x = 0111 1100 1111 1111 1110

f(x)=16, g(x)=2, ans=16+2=18.

1	2	4	8
0	1	1	1
16	32	64	128
1	1	1	1
256	512	1024	2048
1	1	1	1
4096	8192	16384	32768
0	0	1	1
65536	131072	262144	524288
1	1	1	0

(2)

```
#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;
}
```

● 判断题

22. 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。（ ）

23.可能存在输入不同，但输出的第二行相同的情形。（ ）

24.输出的第一行为“-1”。（ ）

● 单选题

25. 设输入字符串长度为 n，decode 函数的时间复杂度为（ ）。

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

26.当输入为“Y3Nx”时，输出的第二行为（ ）。

A. “csp”　　B. “csq”　　C. “CSP”　　D. “Csp”

27. **（3.5 分）**当输入为“Y2NmIDIwMjE=”时，输出的第二行为（ ）。

A. “ccf2021”　　B. “ccf2022”　　C. “ccf 2021”　　D. “ccf 2022”

答案：22-27：✕ √ √ B B C

本题考察的是 base64 的编解码，对于程序中 init()函数的分析，可以得到 base64 表：

base[0]	A	base[26]	a	base[52]	0	table[A]	0
base[1]	B	base[27]	b	base[53]	1	table[B]	1
base[2]	C	base[28]	c	base[54]	2	table[C]	2
base[3]	D	base[29]	d	base[55]	3	table[D]	3
base[4]	E	base[30]	e	base[56]	4	table[E]	4
base[5]	F	base[31]	f	base[57]	5	table[F]	5
base[6]	G	base[32]	g	base[58]	6	...	...
base[7]	H	base[33]	h	base[59]	7	table[Z]	25
base[8]	I	base[34]	i	base[60]	8	table[a]	26
base[9]	J	base[35]	j	base[61]	9	table[b]	27
base[10]	K	base[36]	k	base[62]	+	table[c]	28
base[11]	L	base[37]	l	base[63]	/	table[d]	29
base[12]	M	base[38]	m			table[e]	30
base[13]	N	base[39]	n	上述就是一个		table[f]	31
base[14]	O	base[40]	o	base64 的编码表		...	...
base[15]	P	base[41]	p			table[z]	51
base[16]	Q	base[42]	q			table[0]	52
base[17]	R	base[43]	r	table[0]	0xff	table[1]	53
base[18]	S	base[44]	s	table[1]	0xff	table[2]	54
base[19]	T	base[45]	t	table[2]	0xff	table[3]	55
base[20]	U	base[46]	u	table[3]	0xff	table[4]	56
base[21]	V	base[47]	v	...	...	...	...
base[22]	W	base[48]	w	table[254]	0xff	table[9]	61
base[23]	X	base[49]	x	table[255]	0xff	table[=]	0
base[24]	Y	base[50]	y				
base[25]	Z	base[51]	z				

base64 编码原理

原码	c		s		q	
ASCII 码	99		115		113	
二进制	0110	0011	0111	0011	0111	0001
二进制重组	01 1000		11 0111	00 1101	11 0001	
base64 码	24		55	13	49	
编码(加密)	Y		3	N	x	

逆向则为解密

在线编码解码工具：http://www.jsons.cn/base64/

Base64 编码注意：base64 编码是以 3Byte，也就是 24bit 进行缓存的，如果原码的二进制重组位数不足 24 位，会将空余位补齐 0，也就是 base64 编码字符填充‘=’。

22. 输出是解码，也就是数据的加密前的样子，数据加密前的样子有限制吗，没有。

23.题目是说不同的原码使用同一编码方式会出现同一编码结果，对于 base64 来讲是可以的，因为 base64 会在不足 24bit 时补齐 0，那么如果不补齐 0 也是可以解码出正确的原码的，如：”Y2NmIDIwMjE=”和”Y2NmIDIwMjE”都能解码出同一原码：”ccf 2021”。

24.int(table[0])是将 table[0]转为 int 类型，而 table[0]=0xff，且 table[]是 char 类型，占用 1Byte=8bit，故而：table[0]=1111 1111 B，输出结果为 -1。

26. decode 函数的时间复杂度是由字符串长度控制的，所以为 O(n)。

27.逆向解码过于麻烦，直接带入选项正向编码。

(3)

```
1   #include <iostream>
2   using namespace std;
3
4   const int n = 100000;
5   const int N = n + 1;
6
7   int m;
8   int a[N], b[N], c[N], d[N];
9   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  }
39
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（ ）。

B. $O(n)$　　B. $O(n \log n)$　　C. $O(n\sqrt{n})$　　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（ ）。
A. “15 1340”　　B. “15 2340”　　C. “16 2340”　　D. “16 1340”

答案：28-33：√××ACC
程序意义：欧拉筛法
a[i]=1 表示 i 是合数，a[i]=0 表示 i 是质数。
b[i]表示第 i+1 个质数
c[i]
d[i]
f[i]表示 i 的因子的个数
g[i]表示 i 的因子的和

28.看程序中，f[i],g[i]的取值于 f[i-1]并无联系，f[1]=g[1]=1 只是对 1 的特殊处理。
29.f[i] / c[i * k] = f[i] / (c[i]+1)，代入数据发现是存在倍数关系的
30.因子的个数不一定递增，那么因子的和也不一定递增
31.欧拉筛法的时间复杂度为 O(n)
32.0-100 中因子数为 2 的数据（也就是质数）的个数：25 个
2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53,59,61,67,71,73,79,83,89,97，共 25 个
33.1000=2*2*2*5*5*5,其因子为:1,2,4,5,8,10,20,25,40,50,100,125,200,250,500,1000.
所以 f[i]=16,g[i]=2340

三、完善程序（单选题，每小题 3 分，共计 30 分）
（1）（Josephus 问题）有 n 个人围成一个圈，依次标号 0 至 n - 1。从 0 号开始，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。
试补全模拟程序。

```
1   #include <iostream>
2
3   using namespace std;
4
5   const int MAXN = 1000000;
6   int F[MAXN];
7
8   int main() {
9       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

答案：34-38：D C C D B

34. 出圈人数为 n-1 人

35. 报 0 不出圈，报 1 出圈，故填 p

36. 出圈人数加 1

37. 交替报数

38. 继续报数，并制造一个环，当一轮报数结束，又从头开始。

（2）（矩形计数）平面上有 ？ 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。试补全枚举算法。

```
1   #include <iostream>
2
3   using namespace std;
4
5   struct point {
6       int x, y, id;
7   };
8
```

```
9  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 }
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 == A[j].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

答案：39-43：B D C B D
39.按照矩形的横纵排序。
40.要放入的下一个矩形，一定不等于上一个矩形，且当 t==0 时，第一个矩形可以直接放入。
41.二分查找的模板：mid=(l+r)/2。
42.下一条执行语句为 a=mid+1，证明 A[mid]<p。
43.确定可以组成一个矩形。