

2025 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

认证时间：2025 年 9 月 20 日 09:30~11:30

考生注意事项：

- 试题共有 9 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、电子手表等）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. 一个 32 位无符号整数可以表示的最大值，最接近下列哪个选项？（ ）
A. 4×10^9 B. 3×10^{10} C. 2×10^9 D. 2×10^{10}
2. 在 C++ 中，执行 `int x = 255; cout << (x & (x - 1));` 后，输出的结果是？（ ）
A. 255 B. 254 C. 128 D. 0
3. 函数 `calc(n)` 的定义如下，则 `calc(5)` 的返回值是多少？（ ）

```
int calc(int n) {  
    if (n <= 1) return 1;  
    if (n % 2 == 0) return calc(n / 2) + 1;  
    else return calc(n - 1) + calc(n - 2);  
}
```

- A. 5 B. 6 C. 7 D. 8
4. 用 5 个权值 10, 12, 15, 20, 25 构造哈夫曼树，该树的带权路径长度是多少？（ ）
A. 176 B. 186 C. 196 D. 206
5. 在一个有向图中，所有顶点的入度之和等于所有顶点的出度之和，这个总和等于？（ ）
A. 顶点数 B. 边数 C. 顶点数 + 边数 D. 顶点数 * 2
6. 从 5 位男生和 4 位女生中选出 4 人组成一个学习小组，要求学习小组中男生和女生都有。有多少种不同的选举方法？（ ）
A. 126 B. 121 C. 120 D. 100
7. 假设 `a`, `b`, `c` 都是布尔变量，逻辑表达式 `(a && b) || (!c && a)` 的值与下列哪个表达式不始终相等？（ ）

- A. $a \ \&\& \ (b \ || \ !c)$ B. $(a \ || \ !c) \ \&\& \ (b \ || \ !c) \ \&\& \ (a \ || \ a)$
C. $a \ \&\& \ (!b \ || \ c)$ D. $!(a \ || \ !b) \ || \ (a \ \&\& \ !c)$
8. 已知 $f[0] = 1$, $f[1] = 1$, 并且对于所有 $n \geq 2$ 有 $f[n] = (f[n-1] + f[n-2]) \% 7$. 那么 $f[2025]$ 的值是多少? ()

A. 2 B. 4 C. 5 D. 6

9. 下列关于 C++ string 类的说法, 正确的是? ()

- A. string 对象的长度在创建后不能改变。
B. 可以使用 + 运算符直接连接一个 string 对象和一个 char 类型的字符。
C. string 的 length() 和 size() 方法返回的值可能不同。
D. string 对象必须以 '\0' 结尾, 且这个结尾符计入 length()。

10. 考虑以下 C++ 函数:

```
void solve(int &a, int b) {  
    a = a + b;  
    b = a - b;  
    a = a - b;  
}  
  
int main() {  
    int x = 5, y = 10;  
    solve(x, y);  
}
```

在 main 函数调用 solve 后, x 和 y 的值分别是? ()

A. 5, 10 B. 10, 5 C. 10, 10 D. 5, 5

11. 一个 8×8 的棋盘, 左上角坐标为 (1,1), 右下角为 (8,8)。一个机器人从 (1,1) 出发, 每次只能向右或向下走一格。要到达 (4,5), 有多少种不同的路径? ()

A. 20 B. 35 C. 56 D. 70

12. 某同学用冒泡排序对数组 {6, 1, 5, 2, 4} 进行升序排序, 请问需要进行多少次元素交换? ()

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

13. 十进制数 720_{10} 和八进制数 270_8 的和用十六进制表示是多少? ()

A. 388_{16} B. $3DE_{16}$ C. 288_{16} D. 990_{16}

14. 一棵包含 1000 个结点的完全二叉树, 其叶子结点的数量是多少? ()

A. 499 B. 512 C. 500 D. 501

15. 给定一个初始为空的整数栈 S 和一个空的队列 P。我们按顺序处理输入的整数队列 A: 7, 5, 8, 3, 1, 4, 2。对于队列 A 中的每一个数, 执行以下规则: 如果该数是奇数, 则将其压入栈 S; 如果该数是偶数, 且栈 S 非空, 则弹出一个栈顶元素, 并加入到队列 P 的末尾; 如果该数是偶数, 且栈 S 为空, 则不进行任何操作。当队列 A 中的所有数都处理完毕后, 队列 P 的内容是什么? ()

A. 5, 1, 3 B. 7, 5, 3
C. 3, 1, 5 D. 5, 1, 3, 7

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)

```
01  #include <algorithm>
02  #include <cstdio>
03  #include <cstring>
04  inline int gcd(int a, int b) {
05      if (b == 0)
06          return a;
07      return gcd(b, a % b);
08  }
09  int main() {
10      int n;
11      scanf("%d", &n);
12      int ans = 0;
13      for (int i = 1; i <= n; ++i) {
14          for (int j = i + 1; j <= n; ++j) {
15              for (int k = j + 1; k <= n; ++k) {
16                  if (gcd(i, j) == 1 && gcd(j, k) == 1
17                      && gcd(i, k) == 1) {
18                      ++ans;
19                  }
20              }
21          }
22      }
23      printf("%d\n", ans);
24      return 0;
25  }
```

● 判断题

16. (1 分) 当输入为 2 时, 程序并不会执行第 16 行的判断语句。 ()
17. 将第 16 行中的 “&& gcd(i,k)==1” 删去不会影响程序运行结果。 ()
18. 当输入的 $n \geq 3$ 的时候, 程序总是输出一个正整数。 ()

● 单选题

19. 将第 7 行的 “gcd(b,a%b)” 改为 “gcd(a,a%b)” 后, 程序可能出现的问题是 ()。
- A. 输出的答案大于原答案。 B. 输出的答案小于原答案。
C. 程序有可能陷入死循环。 D. 可能发生整型溢出问题。
20. 当输入为 8 的时候, 输出为 ()。
- A. 37 B. 42 C. 35 D. 25
21. 调用 gcd(36,42) 会返回 ()。
- A. 6 B. 252 C. 3 D. 2

(2)

```
01 #include <algorithm>
02 #include <cstdio>
03 #include <cstring>
04 #define ll long long
05 int n, k;
06 int a[200007];
07 int ans[200007];
08 int main() {
09     scanf("%d%d", &n, &k);
10     for (int i = 1; i <= n; ++i) {
11         scanf("%d", &a[i]);
12     }
13     std::sort(a + 1, a + n + 1);
14     n = std::unique(a + 1, a + n + 1) - a - 1;
15     for (int i = 1, j = 0; i <= n; ++i) {
16         for (; j < i && a[i] - a[j + 1] > k; ++j)
17             ;
18         ans[i] = ans[j] + 1;
19     }
20     printf("%d\n", ans[n]);
```


21	return 0;
22	}

● 判断题

22. 当输入为“3 1 3 2 1”时，输出结果为 2。（ ）

23. 假设输入的 n 为正整数，输出的答案一定小于等于 n ，大于等于 1。（ ）

24. 将第 14 行的“`n=std::unique(a+1,a+n+1)-a-1;`”删去后，有可能出现与原本代码不同的输出结果。（ ）

● 单选题

25. 假设输入的 a 数组和 k 均为正整数，执行第 18 行代码时，一定满足的条件不包括（ ）。

- A. $j \leq i$ B. $a[i] - a[j] > k$ C. $j \leq n$ D. $a[j] < a[i]$

26. 当输入的 $n=100$ 、 $k=2$ 、 $a=\{1, 2, \dots, 100\}$ 时，输出为（ ）。

- A. 34 B. 100 C. 50 D. 33

27. 假设输入的 a 数组和 k 均为正整数，但 a 数组不一定有序，则若误删去第 13 行的

“`std::sort(a+1,a+n+1);`”，程序有可能出现的问题有（ ）。

- A. 输出的答案比原本答案更大 B. 输出的答案比原本答案更小
C. 出现死循环行为 D. 以上均可能发生

(3)

01	#include <algorithm>
02	#include <cstdio>
03	#include <cstring>
04	#define ll long long
05	int f[5007][5007];
06	int a[5007], b[5007];
07	int n;
08	int main() {
09	scanf("%d", &n);
10	for (int i = 1; i <= n; ++i) {
11	scanf("%d", &a[i]);
12	}
13	for (int i = 1; i <= n; ++i) {
14	scanf("%d", &b[i]);
15	}

```

16 for (int i = 1; i <= n; ++i) {
17     for (int j = 1; j <= n; ++j) {
18         f[i][j] = std::max(f[i][j], std::max(f[i - 1][j], f[i][j - 1]));
19         if (a[i] == b[j]) {
20             f[i][j] = std::max(f[i][j], f[i - 1][j - 1] + 1);
21         }
22     }
23 }
24 printf("%d\n", f[n][n]);
25 return 0;
26 }

```

● 判断题

28. 当输入“4 1 2 3 4 1 3 2 2”时，输出为 2。 ()
29. 当程序运行完毕后，对于所有的 $1 \leq i, j \leq n$ ，都一定有 $f[i][j] \leq f[n][n]$ 。 ()
30. 将第 18 行的“ $f[i][j] = \text{std::max}(f[i][j], \text{std::max}(f[i-1][j], f[i][j-1]));$ ”删去后，并不影响程序运行结果。 ()

● 单选题

31. 输出的答案满足的性质有 ()。
- A. 小于等于 n B. 大于等于 0 C. 不一定大于等于 1 D. 以上均是
32. 如果在 16 行的循环前加上以下两行：“ $\text{std::sort}(a+1, a+n+1);$
 $\text{std::sort}(b+1, b+n+1);$ ”，则答案会 ()。
- A. 变大或不变 B. 变小或不变 C. 一定变大 D. 不变
33. 如果输入的 $a = \{1, 2, \dots, n\}$ ，而且 b 数组中数字均为 $1 \sim n$ 中的正整数，则上述代码等价于下面哪个问题： ()。
- A. 求 b 数组去重后的长度 B. 求 b 数组的最长上升子序列
 C. 求 b 数组的长度 D. 求 b 数组的最大值

三、完善程序 (单选题，每小题 3 分，共计 30 分)

(1) (字符串解码) “行程长度编码” (Run-Length Encoding) 是一种无损压缩算法，常用于压缩重复字符较多的数据，以减少存储空间。假设原始字符串不包含数字字符。压缩规则如下：i) 如果原始字符串中一个字符连续出现 N 次 ($N \geq 2$)，在压缩字符串中它被表示为“字

符+数字 N”。例如，编码“A12”代表 12 个连续的字符 A。ii) 如果原始字符串中一个字符只出现 1 次，在压缩字符串中它就表示为该字符本身。例如，编码“B”代表 1 个字符 B。

以下程序实现读取压缩字符串并输出其原始的、解压后的形式。试补全程序。

```
01 #include <cctype>
02 #include <iostream>
03 #include <string>
04 using namespace std;
05
06 int main() {
07     string z;
08     cin >> z;
09     string s = "";
10
11     for (int i = 0; i < z.length(); ) {
12         char ch = z[i];
13
14         if (____①____ && isdigit(z[i + 1])) {
15             i++;
16             int count = 0;
17             while (i < z.length() && isdigit(z[i])) {
18                 count = ____②____;
19                 i++;
20             }
21             for (int j = 0; j < ____③____; ++j) {
22                 s += ch;
23             }
24         } else {
25             s += ____④____;
26             ____⑤____;
27         }
28     }
29
30     cout << s << endl;
31     return 0;
32 }
```

33. ①处应填 ()

- | | |
|---------------------------------------|-------------------------------|
| A. <code>i < z.length()</code> | B. <code>i - 1 >= 0</code> |
| C. <code>i + 1 < z.length()</code> | D. <code>isdigit(z[i])</code> |

34. ②处应填 ()

A. `count + (z[i] - '0')`

B. `count * 10 + (z[i] - '0')`

C. `z[i] - '0'`

D. `count + 1`

35. ③处应填 ()

A. `count - 1`

B. `count`

C. `10`

D. `z[i] - '0'`

36. ④处应填 ()

A. `z[i+1]`

B. `ch`

C. `z.back()`

D. `(char)z[i] + 1`

37. ⑤处应填 ()

A. `i--`

B. `i = i + 2`

C. `i++`

D. `// 不执行任何操作`

(2) (精明与糊涂) 有 N 个人，分为两类：i) 精明人：永远能正确判断其他人是精明还是糊涂；ii) 糊涂人：判断不可靠，会给出随机的判断。已知精明人严格占据多数，即如果精明人有 k 个，则满足 $k > N/2$ 。

你只能通过函数 `query(i,j)` 让第 i 个人判断第 j 个人：返回 `true` 表示判断结果为“精明人”；返回 `false` 表示判断结果为“糊涂人”。你的目标是，通过这些互相判断，找出至少一个百分之百能确定的精明人。同时，你无需关心 `query(i,j)` 的内部实现。

以下程序利用“精明人占多数”的优势。设想一个“消除”的过程，让人们互相判断并进行抵消。经过若干轮抵消后，最终留下的候选者必然属于多数派，即精明人。

例如，假设有三个人 0、1、2。如果 0 说 1 是糊涂人，而 1 也说 0 是糊涂人，则 0 和 1 至少有一个是糊涂人。程序将同时淘汰 0 和 1。由于三人里至少有两个精明人，我们确定 2 是精明人。试补全程序。

```
01 #include <iostream>
02 #include <vector>
03 using namespace std;
04
05 int N;
06 bool query(int i, int j);
07
08 int main() {
09     cin >> N;
10
11     int candidate = 0;
12     int count = ①;
```



```

13
14     for (int i = 1; i < N; ++i) {
15         if (②) {
16             candidate = i;
17             count = 1;
18         } else {
19             if (③) {
20                 ④;
21             } else {
22                 count++;
23             }
24         }
25     }
26
27     cout << ⑤ << endl;
28     return 0;
29 }

```

38. ①处应填 ()

A. 0

B. 1

C. N

D. -1

39. ②处应填 ()

A. count < 0

B. count == 1

C. count == 0

D. query(candidate, i) == false

40. ③处应填 ()

A. query(candidate, i) == false

B. query(i, candidate) == true

C. query(candidate, i) == false && query(i, candidate) == false

D. query(candidate, i) == false || query(i, candidate) == false

41. ④处应填 ()

A. count--

B. break

C. count++

D. candidate = i

42. ⑤处应填 ()

A. N - 1

B. count

C. candidate

D. 0