

# 2022 CCF 非专业级别软件能力认证第一轮

## (CSP-S1) 提高级 C++语言试题

认证时间：2022 年 9 月 18 日 14:30~16:30

### 考生注意事项：

- 试题纸共有 13 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. 在 Linux 系统终端中，用于切换工作目录的命令为（ ）。

- A. ls
- B. cd
- C. cp
- D. all

2. 你同时用 `time` 命令和秒表为某个程序在单核 CPU 的运行计时。假如 `time` 命令的输出如下：

```
real    0m30.721s
user    0m24.579s
sys     0m6.123s
```

以下最接近秒表计时的时长为（ ）。

- A. 30s
- B. 24s
- C. 18s
- D. 6s

3. 若元素 a、b、c、d、e、f 依次进栈，允许进栈、退栈操作交替进行，但不允许连续三次退栈操作，则不可能得到的出栈序列是（ ）。

- A. dcebf a
- B. cbdaef
- C. bcaefd

D. afedcb

4. 考虑对  $n$  个数进行排序，以下最坏时间复杂度低于  $O(n^2)$  的排序方法是（ ）。
- A. 插入排序
  - B. 冒泡排序
  - C. 归并排序
  - D. 快速排序
5. 假设在基数排序过程中，受宇宙射线的影响，某项数据异变为一个完全不同的值。请问排序算法结束后，可能出现的最坏情况是（ ）。
- A. 移除受影响的数据后，最终序列是有序序列
  - B. 移除受影响的数据后，最终序列是前后两个有序的子序列
  - C. 移除受影响的数据后，最终序列是一个有序的子序列和一个基本无序的子序列
  - D. 移除受影响的数据后，最终序列基本无序
6. 计算机系统用小端（Little Endian）和大端（Big Endian）来描述多字节数据的存储地址顺序模式，其中小端表示将低位字节数据存储在低地址的模式、大端表示将高位字节数据存储在低地址的模式。在小端模式的系统和大端模式的系统分别编译和运行以下 C++ 代码段表示的程序，将分别输出什么结果？（ ）
- ```
unsigned x = 0xDEADBEEF;
unsigned char *p = (unsigned char *)&x;
printf("%X", *p);
```
- A. EF、EF
  - B. EF、DE
  - C. DE、EF
  - D. DE、DE
7. 一个深度为 5（根结点深度为 1）的完全 3 叉树，按前序遍历的顺序给结点从 1 开始编号，则第 100 号结点的父结点是第（ ）号。

- A. 95
- B. 96
- C. 97
- D. 98

8. 强连通图的性质不包括 ( ):

- A. 每个顶点的度数至少为 1
- B. 任意两个顶点之间都有边相连
- C. 任意两个顶点之间都有路径相连
- D. 每个顶点至少都连有一条边

9. 每个顶点度数均为 2 的无向图称为“2 正规图”。由编号为从 1 到  $n$  的顶点构成的所有 2 正规图, 其中包含欧拉回路的不同 2 正规图的数量为 ( )。

- A.  $n!$
- B.  $(n-1)!$
- C.  $n!/2$
- D.  $(n-1)!/2$

10. 共有 8 人选修了程序设计课程, 期末大作业要求由 2 人组成的团队完成。假设不区分每个团队内 2 人的角色和作用, 请问共有多少种可能的组队方案。( )

- A. 28
- B. 32
- C. 56
- D. 64

11. 小明希望选到形如“省 A •  $\mathcal{L} \mathcal{L} \mathcal{D} \mathcal{D} \mathcal{D}$ ”的车牌号。车牌号在“•”之前的内容固定不变; 后面的 5 位号码中, 前 2 位必须是大写英文字母, 后 3 位必须是阿拉伯数字 ( $\mathcal{L}$  代表 A 至 Z,  $\mathcal{D}$  表示 0 至 9, 两个  $\mathcal{L}$  和三个  $\mathcal{D}$  之间可能相同也可能不同)。请问总共有多少个可供选择的车牌号。( )

- A. 20280
- B. 52000
- C. 676000
- D. 1757600

12. 给定地址区间为 0~9 的哈希表，哈希函数为  $h(x) = x \% 10$ ，采用线性探查的冲突解决策略（对于出现冲突情况，会往后探查第一个空的地址存储；若地址 9 冲突了则从地址 0 重新开始探查）。哈希表初始为空表，依次存储(71, 23, 73, 99, 44, 79, 89)后，请问 89 存储在哈希表哪个地址中。（ ）

- A. 9
- B. 0
- C. 1
- D. 2

13. 对于给定的  $n$ ，分析以下代码段对应的时间复杂度，其中最为准确的时间复杂度为（ ）。

```
int i, j, k = 0;
for (i = 0; i < n; i++) {
    for (j = 0; j < n; j*=2) {
        k = k + n / 2;
    }
}
```

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

14. 以比较为基本运算，在  $n$  个数的数组中找最大的数，在最坏情况下至少要做（ ）次运算。

- A.  $n/2$
- B.  $n-1$

- C. n
- D. n+1

15. ack 函数在输入参数“(2,2)”时的返回值为( )。

```
unsigned ack(unsigned m, unsigned n) {  
    if (m == 0) return n + 1;  
    if (n == 0) return ack(m - 1, 1);  
    return ack(m - 1, ack(m, n - 1));  
}
```

- A. 5
- B. 7
- C. 9
- D. 13

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 <iostream>  
02 #include <string>  
03 #include <vector>  
04  
05 using namespace std;  
06  
07 int f(const string &s, const string &t)  
08 {  
09     int n = s.length(), m = t.length();  
10  
11     vector<int> shift(128, m + 1);  
12  
13     int i, j;  
14  
15     for (j = 0; j < m; j++)  
16         shift[t[j]] = m - j;  
17  
18     for (i = 0; i <= n - m; i += shift[s[i + m]]) {
```

```

19         j = 0;
20         while (j < m && s[i + j] == t[j]) j++;
21         if (j == m) return i;
22     }
23
24     return -1;
25 }
26
27 int main()
28 {
29     string a, b;
30     cin >> a >> b;
31     cout << f(a, b) << endl;
32     return 0;
33 }

```

假设输入字符串由 **ASCII** 可见字符组成，完成下面的判断题和单选题：

● 判断题

16. (1 分) 当输入为 “abcde fg” 时，输出为 -1。 ( )
17. 当输入为 “abbababbbab abab” 时，输出为 4。 ( )
18. 当输入为 “GoodLuckCsp2022 22” 时，第 20 行的 “j++” 语句执行次数为 2。 ( )

● 单选题

19. 该算法最坏情况下的时间复杂度为 ( )。
- A.  $O(n + m)$       B.  $O(n \log m)$       C.  $O(m \log n)$       D.  $O(nm)$
20.  $f(a, b)$  与下列 ( ) 语句的功能最类似。
- A. `a.find(b)`      B. `a.rfind(b)`  
 C. `a.substr(b)`      D. `a.compare(b)`
21. 当输入为 “baaabaaabaaabaaaa aaaa”，第 20 行的 “j++” 语句执行次数为 ( )。
- A. 9      B. 10      C. 11      D. 12

(2)

```

01 #include <iostream>
02
03 using namespace std;
04
05 const int MAXN = 105;
06
07 int n, m, k, val[MAXN];
08 int temp[MAXN], cnt[MAXN];

```

```

09
10 void init()
11 {
12     cin >> n >> k;
13     for (int i = 0; i < n; i++) cin >> val[i];
14     int maximum = val[0];
15     for (int i = 1; i < n; i++)
16         if (val[i] > maximum) maximum = val[i];
17     m = 1;
18     while (maximum >= k) {
19         maximum /= k;
20         m++;
21     }
22 }
23
24 void solve()
25 {
26     int base = 1;
27     for (int i = 0; i < m; i++) {
28         for (int j = 0; j < k; j++) cnt[j] = 0;
29         for (int j = 0; j < n; j++) cnt[val[j] / base % k]++;
30         for (int j = 1; j < k; j++) cnt[j] += cnt[j - 1];
31         for (int j = n - 1; j >= 0; j--) {
32             temp[cnt[val[j] / base % k] - 1] = val[j];
33             cnt[val[j] / base % k]--;
34         }
35         for (int j = 0; j < n; j++) val[j] = temp[j];
36         base *= k;
37     }
38 }
39
40 int main()
41 {
42     init();
43     solve();
44     for (int i = 0; i < n; i++) cout << val[i] << ' ';
45     cout << endl;
46     return 0;
47 }

```

假设输入的  $n$  为不大于 **100** 的正整数， $k$  为不小于 **2** 且不大于 **100** 的正整数， $val[i]$  在 `int` 表示范围内，完成下面的判断题和单选题：

● 判断题

22. 这是一个不稳定的排序算法。（    ）

23. 该算法的空间复杂度仅与  $n$  有关。 ( )

24. 该算法的时间复杂度为  $O(m(n+k))$ 。 ( )

● 单选题

25. 当输入为“5 3 98 26 91 37 46”时，程序第一次执行到第 36 行，`val[]`数组的内容依次为 ( )。

A. 91 26 46 37 98

B. 91 46 37 26 98

C. 98 26 46 91 37

D. 91 37 46 98 26

26. 若 `val[i]` 的最大值为 100， $k$  取 ( ) 时算法运算次数最少。

A. 2

B. 3

C. 10

D. 不确定

27. 当输入的  $k$  比 `val[i]` 的最大值还大时，该算法退化为 ( ) 算法。

A. 选择排序

B. 冒泡排序

C. 计数排序

D. 桶排序

(3)

```
01 #include <iostream>
02 #include <algorithm>
03
04 using namespace std;
05
06 const int MAXL = 1000;
07
08 int n, k, ans[MAXL];
09
10 int main(void)
11 {
12     cin >> n >> k;
13     if (!n) cout << 0 << endl;
14     else
15     {
16         int m = 0;
17         while (n)
18         {
19             ans[m++] = (n % (-k) + k) % k;
20             n = (ans[m - 1] - n) / k;
21         }
22         for (int i = m - 1; i >= 0; i--)
23             cout << char(ans[i] >= 10 ?
24                 ans[i] + 'A' - 10 :
25                 ans[i] + '0');
26         cout << endl;
```



```

27     }
28     return 0;
29 }

```

假设输入的  $n$  在 `int` 范围内， $k$  为不小于 2 且不大于 36 的正整数，完成下面的判断题和单选题：

● 判断题

28. 该算法的时间复杂度为  $O(\log_k n)$ 。 ( )

29. 删除第 23 行的强制类型转换，程序的行为不变。 ( )

30. 除非输入的  $n$  为 0，否则程序输出的字符数为  $O(\lceil \log_k |n| \rceil + 1)$ 。 ( )

● 单选题

31. 当输入为 “100 7” 时，输出为 ( )。

- A. 202                      B. 1515                      C. 244                      D. 1754

32. 当输入为 “-255 8” 时，输出为 “ ( ) ”。

- A. 1400                      B. 1401                      C. 417                      D. 400

33. 当输入为 “1000000 19” 时，输出为 “ ( ) ”。

- A. BG939                      B. 87GIB  
C. 1CD428                      D. 7CF1B

### 三、 完善程序（单选题，每小题 3 分，共计 30 分）

(1)（归并第  $k$  小）已知两个长度均为  $n$  的有序数组  $a1$  和  $a2$ （均为递增序，但不保证严格单调递增），并且给定正整数  $k$  ( $1 \leq k \leq 2n$ )，求数组  $a1$  和  $a2$  归并排序后的数组里第  $k$  小的数值。

试补全程序。

```

01 #include <bits/stdc++.h>
02 using namespace std;
03
04 int solve(int *a1, int *a2, int n, int k) {
05     int left1 = 0, right1 = n - 1;
06     int left2 = 0, right2 = n - 1;
07     while (left1 <= right1 && left2 <= right2) {
08         int m1 = (left1 + right1) >> 1;
09         int m2 = (left2 + right2) >> 1;
10         int cnt = ①;
11         if (②) {

```

```

12     if (cnt < k) left1 = m1 + 1;
13     else right2 = m2 - 1;
14 } else {
15     if (cnt < k) left2 = m2 + 1;
16     else right1 = m1 - 1;
17 }
18 }
19 if (③) {
20     if (left1 == 0) {
21         return a2[k - 1];
22     } else {
23         int x = a1[left1 - 1], ④;
24         return std::max(x, y);
25     }
26 } else {
27     if (left2 == 0) {
28         return a1[k - 1];
29     } else {
30         int x = a2[left2 - 1], ⑤;
31         return std::max(x, y);
32     }
33 }
34 }

```

34. ①处应填 ( )

- |                    |                          |
|--------------------|--------------------------|
| A. $(m1 + m2) * 2$ | B. $(m1 - 1) + (m2 - 1)$ |
| C. $m1 + m2$       | D. $(m1 + 1) + (m2 + 1)$ |

35. ②处应填 ( )

- |                       |                       |
|-----------------------|-----------------------|
| A. $a1[m1] == a2[m2]$ | B. $a1[m1] <= a2[m2]$ |
| C. $a1[m1] >= a2[m2]$ | D. $a1[m1] != a2[m2]$ |

36. ③处应填 ( )

- |                      |                      |
|----------------------|----------------------|
| A. $left1 == right1$ | B. $left1 < right1$  |
| C. $left1 > right1$  | D. $left1 != right1$ |

37. ④处应填 ( )

- |                            |                        |
|----------------------------|------------------------|
| A. $y = a1[k - left2 - 1]$ | B. $y = a1[k - left2]$ |
| C. $y = a2[k - left1 - 1]$ | D. $y = a2[k - left1]$ |

38. ⑤处应填 ( )

- |                            |                        |
|----------------------------|------------------------|
| A. $y = a1[k - left2 - 1]$ | B. $y = a1[k - left2]$ |
| C. $y = a2[k - left1 - 1]$ | D. $y = a2[k - left1]$ |

(2) (容器分水) 有两个容器, 容器 1 的容量为  $a$  升, 容器 2 的容量为  $b$  升; 同时允许下列的三种操作, 分别为:

1) FILL( $i$ ): 用水龙头将容器  $i$  ( $i \in \{1, 2\}$ ) 灌满水;

2) DROP( $i$ ): 将容器  $i$  的水倒进下水道;

3) POUR( $i, j$ ): 将容器  $i$  的水倒进容器  $j$  (完成此操作后, 要么容器  $j$  被灌满, 要么容器  $i$  被清空)。

求只使用上述的两个容器和三种操作, 获得恰好  $c$  升水的最少操作数和操作序列。上述  $a$ 、 $b$ 、 $c$  均为不超过 100 的正整数, 且  $c \leq \max\{a, b\}$ 。

试补全程序。

```
01 #include <bits/stdc++.h>
02 using namespace std;
03 const int N = 110;
04
05 int f[N][N];
06 int ans;
07 int a, b, c;
08 int init;
09
10 int dfs(int x, int y) {
11     if (f[x][y] != init)
12         return f[x][y];
13     if (x == c || y == c)
14         return f[x][y] = 0;
15     f[x][y] = init - 1;
16     f[x][y] = min(f[x][y], dfs(a, y) + 1);
17     f[x][y] = min(f[x][y], dfs(x, b) + 1);
18     f[x][y] = min(f[x][y], dfs(0, y) + 1);
19     f[x][y] = min(f[x][y], dfs(x, 0) + 1);
20     int t = min(a - x, y);
21     f[x][y] = min(f[x][y], ①);
22     t = min(x, b - y);
23     f[x][y] = min(f[x][y], ②);
24     return f[x][y];
25 }
26
27 void go(int x, int y) {
28     if (③)
29         return;
30     if (f[x][y] == dfs(a, y) + 1) {
31         cout << "FILL(1)" << endl;
32         go(a, y);
33     } else if (f[x][y] == dfs(x, b) + 1) {
```

```

34     cout << "FILL(2)" << endl;
35     go(x, b);
36 } else if (f[x][y] == dfs(0, y) + 1) {
37     cout << "DROP(1)" << endl;
38     go(0, y);
39 } else if (f[x][y] == dfs(x, 0) + 1) {
40     cout << "DROP(2)" << endl;
41     go(x, 0);
42 } else {
43     int t = min(a - x, y);
44     if (f[x][y] == ④) {
45         cout << "POUR(2,1)" << endl;
46         go(x + t, y - t);
47     } else {
48         t = min(x, b - y);
49         if (f[x][y] == ⑤) {
50             cout << "POUR(1,2)" << endl;
51             go(x - t, y + t);
52         } else
53             assert(0);
54     }
55 }
56 }
57
58 int main() {
59     cin >> a >> b >> c;
60     ans = 1 << 30;
61     memset(f, 127, sizeof f);
62     init = **f;
63     if ((ans = dfs(0, 0)) == init - 1)
64         cout << "impossible";
65     else {
66         cout << ans << endl;
67         go(0, 0);
68     }
69 }

```

39. ①处应填 ( )

- |                                   |                                   |
|-----------------------------------|-----------------------------------|
| A. $\text{dfs}(x + t, y - t) + 1$ | B. $\text{dfs}(x + t, y - t) - 1$ |
| C. $\text{dfs}(x - t, y + t) + 1$ | D. $\text{dfs}(x - t, y + t) - 1$ |

40. ②处应填 ( )

- |                                   |                                   |
|-----------------------------------|-----------------------------------|
| A. $\text{dfs}(x + t, y - t) + 1$ | B. $\text{dfs}(x + t, y - t) - 1$ |
| C. $\text{dfs}(x - t, y + t) + 1$ | D. $\text{dfs}(x - t, y + t) - 1$ |

41. ③处应填 ( )

- A. `x == c || y == c`
- C. `x >= c || y >= c`

- B. `x == c && y == c`
- D. `x >= c && y >= c`

42. ④处应填 ( )

- A. `dfs(x + t, y - t) + 1`
- C. `dfs(x - t, y + t) + 1`

- B. `dfs(x + t, y - t) - 1`
- D. `dfs(x - t, y + t) - 1`

43. ⑤处应填 ( )

- A. `dfs(x + t, y - t) + 1`
- C. `dfs(x - t, y + t) + 1`

- B. `dfs(x + t, y - t) - 1`
- D. `dfs(x - t, y + t) - 1`