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

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

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

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

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

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

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

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

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

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

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

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

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

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（ ）。

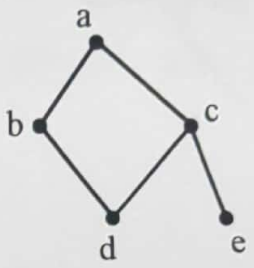
A. 105

B. 840

C. 210

D. 420

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

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

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 <stdio.h>
02
03 int n;
```

```
04 int a[1000];
05
06 int f(int x)
07 {
08     int ret = 0;
09     for (; x; x &= x - 1) ret++;
10     return ret;
11 }
12
13 int g(int x)
14 {
15     return x & -x;
16 }
17
18 int main()
19 {
20     scanf("%d", &n);
21     for (int i = 0; i < n; i++) scanf("%d", &a[i]);
22     for (int i = 0; i < n; i++)
23         printf("%d ", f(a[i]) + g(a[i]));
24     printf("\n");
25     return 0;
26 }
```

- 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（13-16 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”

（2）

```
01 #include <stdio.h>
```

```
02 #include <string.h>
03
04 char base[64];
05 char table[256];
06 char str[256];
07 char ans[256];
08
09 void init()
10 {
11     for (int i = 0; i < 26; i++) base[i] = 'A' + i;
12     for (int i = 0; i < 26; i++) base[26 + i] = 'a' + i;
13     for (int i = 0; i < 10; i++) base[52 + i] = '0' + i;
14     base[62] = '+', base[63] = '/';
15
16     for (int i = 0; i < 256; i++) table[i] = 0xff;
17     for (int i = 0; i < 64; i++) table[base[i]] = i;
18     table['='] = 0;
19 }
20
21 void decode(char *str)
22 {
23     char *ret = ans;
24     int i, len = strlen(str);
25     for (i = 0; i < len; i += 4) {
26         (*ret++) = table[str[i]] << 2 | table[str[i + 1]] >> 4;
27         if (str[i + 2] != '=')
28             (*ret++) = (table[str[i + 1]] & 0x0f) << 4 |
                                        table[str[i + 2]] >> 2;
29         if (str[i + 3] != '=')
30             (*ret++) = table[str[i + 2]] << 6 | table[str[i + 3]];
31     }
32 }
33
34 int main()
35 {
36     init();
37     printf("%d\n", (int)table[0]);
38
39     scanf("%s", str);
40     decode(str);
41     printf("%s\n", ans);
42     return 0;
43 }
```

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

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”

（3）

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

```
26                  g[i * k] = g[i] * k + d[i];
27                  break;
28              }
29              else {
30                  c[i * k] = 1;
31                  f[i * k] = 2 * f[i];
32                  d[i * k] = g[i];
33                  g[i * k] = g[i] * (k + 1);
34              }
35          }
36      }
37  }
38
39  int main()
40  {
41      init();
42
43      int x;
44      scanf("%d", &x);
45      printf("%d %d\n", f[x], g[x]);
46      return 0;
47  }
```

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 12 行删去不会影响输出的结果。（ ）

29.（2 分）第 24 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. $\Theta(n)$　　B. $\Theta(n \log n)$　　C. $\Theta(n\sqrt{n})$　　D. $\Theta(n^2)$

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”

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。

```
01 #include <stdio.h>
02
03 const int MAXN = 1000000;
04 int F[MAXN];
05
06 int main() {
07     int n;
08     scanf("%d", &n);
09     int i = 0, p = 0, c = 0;
10     while (①) {
11         if (F[i] == 0) {
12             if (②) {
13                 F[i] = 1;
14                 ③;
15             }
16             ④;
17         }
18         ⑤;
19     }
20     int ans = -1;
21     for (i = 0; i < n; i++)
22         if (F[i] == 0)
23             ans = i;
24     printf("%d\n", ans);
25     return 0;
26 }
```

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

（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。

试补全枚举算法。

```
01 #include <stdio.h>
02
03 struct point {
04     int x, y, id;
05 };
06
07 int equals(struct point a, struct point b) {
08     return a.x == b.x && a.y == b.y;
09 }
10
11 int cmp(struct point a, struct point b) {
12     return ①;
13 }
14
15 void sort(struct point A[], int n) {
16     for (int i = 0; i < n; i++)
17         for (int j = 1; j < n; j++)
18             if (cmp(A[j], A[j - 1])) {
19                 struct point t = A[j];
20                 A[j] = A[j - 1];
21                 A[j - 1] = t;
22             }
23 }
24
25 int unique(struct point A[], int n) {
26     int t = 0;
27     for (int i = 0; i < n; i++)
28         if (②)
```

```
29              A[t++] = A[i];
30      return t;
31  }
32
33  int binary_search(struct point A[], int n, int x, int y) {
34      struct point p;
35      p.x = x;
36      p.y = y;
37      p.id = n;
38      int a = 0, b = n - 1;
39      while (a < b) {
40          int mid = ③;
41          if (④)
42              a = mid + 1;
43          else
44              b = mid;
45      }
46      return equals(A[a], p);
47  }
48
49  #define MAXN 1000
50  struct point A[MAXN];
51
52  int main() {
53      int n;
54      scanf("%d", &n);
55      for (int i = 0; i < n; i++) {
56          scanf("%d %d", &A[i].x, &A[i].y);
57          A[i].id = i;
58      }
59      sort(A, n);
60      n = unique(A, n);
61      int ans = 0;
62      for (int i = 0; i < n; i++)
63          for (int j = 0; j < n; j++)
64              if (⑤ && binary_search(A, n, A[i].x, A[j].y) &&
                        binary_search(A, n, A[j].x, A[i].y)) {
65                  ans++;
66              }
67      printf("%d\n", ans);
68      return 0;
69  }
```

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`