2022 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：2022 年 9 月 18 日 09:30~11:30

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．以下哪种功能没有涉及 C++语言的面向对象特性支持：（ ）。

A．C++中调用 printf 函数

B．C++中调用用户定义的类成员函数

C．C++中构造一个 class 或 struct

D．C++中构造来源于同一基类的多个派生类

2．有 6 个元素，按照 6、5、4、3、2、1 的顺序进入栈 S，请问下列哪个出栈序列是非法的（ ）。

A．5 4 3 6 1 2

B．4 5 3 1 2 6

C．3 4 6 5 2 1

D．2 3 4 1 5 6

3．运行以下代码片段的行为是（ ）。

```
int x = 101;
int y = 201;
int *p = &x;
int *q = &y;
p = q;
```

A．将 x 的值赋为 201

B．将 y 的值赋为 101

公众号：黑猫编程
网址：https://noi.hioier.com

C．将 q 指向 x 的地址

D．将 p 指向 y 的地址

4．链表和数组的区别包括（　）。

A．数组不能排序，链表可以

B．链表比数组能存储更多的信息

C．数组大小固定，链表大小可动态调整

D．以上均正确

5．对假设栈 S 和队列 Q 的初始状态为空。存在 e1~e6 六个互不相同的数据，每个数据按照进栈 S、出栈 S、进队列 Q、出队列 Q 的顺序操作，不同数据间的操作可能会交错。已知栈 S 中依次有数据 e1、e2、e3、e4、e5 和 e6 进栈，队列 Q 依次有数据 e2、e4、e3、e6、e5 和 e1 出队列。则栈 S 的容量至少是（　）个数据。

A．2

B．3

C．4

D．6

6．对表达式 a+(b-c)*d 的前缀表达式为（　），其中+、-、*是运算符。

A．*+a-bcd

B．+a*-bcd

C．abc-d*+

D．abc-+d

7．假设字母表 {a，b，c，d，e} 在字符串出现的频率分别为 10%，15%，30%，16%，29%。若使用哈夫曼编码方式对字母进行不定长的二进制编码，字母 d 的编码长度为（　）位。

A．1

B．2

公众号：黑猫编程
网址：https://noi.hioier.com

C．2 或 3

D．3

8．一棵有 n 个结点的完全二叉树用数组进行存储与表示，已知根结点存储在数组的第 1 个位置。若存储在数组第 9 个位置的结点存在兄弟结点和两个子结点，则它的兄弟结点和右子结点的位置分别是（　）。

A．8、18

B．10、18

C．8、19

D．10、19

9．考虑由 N 个顶点构成的有向连通图，采用邻接矩阵的数据结构表示时，该矩阵中至少存在（　）个非零元素。

A．N-1

B．N

C．N+1

D．N^2

10．以下对数据结构的表述不恰当的一项为：（　）。

A．图的深度优先遍历算法常使用的数据结构为栈。

B．栈的访问原则为后进先出，队列的访问原则是先进先出。

C．队列常常被用于广度优先搜索算法。

D．栈与队列存在本质不同，无法用栈实现队列。

11．以下哪组操作能完成在双向循环链表结点 p 之后插入结点 s 的效果（其中，next 域为结点的直接后继，prev 域为结点的直接前驱）：（　）。

A．p->next->prev=s;　s->prev=p;　p->next=s;　s->next=p->next;

B．p->next->prev=s;　p->next=s;　s->prev=p;　s->next=p->next;

C．s->prev=p;　s->next=p->next;　p->next=s;　p->next->prev=s;

公众号：黑猫编程
网址：https://noi.hioier.com

D. s->next=p->next; p->next->prev=s; s->prev=p; p->next=s;

12. 以下排序算法的常见实现中，哪个选项的说法是错误的：（ ）。

A. 冒泡排序算法是稳定的

B. 简单选择排序是稳定的

C. 简单插入排序是稳定的

D. 归并排序算法是稳定的

13. 八进制数 32.1 对应的十进制数是（ ）。

A. 24.125

B. 24.250

C. 26.125

D. 26.250

14. 一个字符串中任意个连续的字符组成的子序列称为该字符串的子串，则字符串 abcab 有（ ）个内容互不相同的子串。

A. 12

B. 13

C. 14

D. 15

15. 以下对递归方法的描述中，正确的是：（ ）

A. 递归是允许使用多组参数调用函数的编程技术

B. 递归是通过调用自身来求解问题的编程技术

C. 递归是面向对象和数据而不是功能和逻辑的编程语言模型

D. 递归是将用某种高级语言转换为机器代码的编程技术

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）

公众号：黑猫编程
网址：https://noi.hioier.com

(1)

```
01 #include <iostream>
02
03 using namespace std;
04
05 int main()
06 {
07     unsigned short x, y;
08     cin >> x >> y;
09     x = (x | x << 2) & 0x33;
10     x = (x | x << 1) & 0x55;
11     y = (y | y << 2) & 0x33;
12     y = (y | y << 1) & 0x55;
13     unsigned short z = x | y << 1;
14     cout << z << endl;
15     return 0;
16 }
```

假设输入的 x、y 均是不超过 15 的自然数，完成下面的判断题和单选题：

- 判断题

16. 删去第 7 行与第 13 行的 unsigned，程序行为不变。（ ）

17. 将第 7 行与第 13 行的 short 均改为 char，程序行为不变。（ ）

18. 程序总是输出一个整数“0”。（ ）

19. 当输入为“2 2”时，输出为“10”。（ ）

20. 当输入为“2 2”时，输出为“59”。（ ）

- 单选题

21. 当输入为“13 8”时，输出为（ ）。

A. “0”　　B. “209”　　C. “197”　　D. “226”

（2）

```
01 #include <algorithm>
02 #include <iostream>
03 #include <limits>
04
05 using namespace std;
06
07 const int MAXN = 105;
```


公众号：黑猫编程
网址：https://noi.hioier.com

```
08 const int MAXK = 105;
09
10 int h[MAXN][MAXK];
11
12 int f(int n, int m)
13 {
14     if (m == 1) return n;
15     if (n == 0) return 0;
16
17     int ret = numeric_limits<int>::max();
18     for (int i = 1; i <= n; i++)
19         ret = min(ret, max(f(n - i, m), f(i - 1, m - 1)) + 1);
20     return ret;
21 }
22
23 int g(int n, int m)
24 {
25     for (int i = 1; i <= n; i++)
26         h[i][1] = i;
27     for (int j = 1; j <= m; j++)
28         h[0][j] = 0;
29
30     for (int i = 1; i <= n; i++) {
31         for (int j = 2; j <= m; j++) {
32             h[i][j] = numeric_limits<int>::max();
33             for (int k = 1; k <= i; k++)
34             h[i][j] = min(
35
36                 h[i][j],
                    max(h[i - k][j], h[k - 1][j - 1]) + 1);
37         }
38     }
39
40     return h[n][m];
41 }
42
43 int main()
44 {
45     int n, m;
46     cin >> n >> m;
47     cout << f(n, m) << endl << g(n, m) << endl;
48     return 0;
49 }
```

假设输入的 n、m 均是不超过 100 的正整数，完成下面的判断题和单选题：

公众号：黑猫编程
网址：https://noi.hioier.com

- 判断题

22.当输入为“7 3”时，第 19 行用来取最小值的 min 函数执行了 449 次。（ ）

23.输出的两行整数总是相同的。（ ）

24.当 m 为 1 时，输出的第一行总为 n。（ ）

- 单选题

25.算法 g(n,m)最为准确的时间复杂度分析结果为（ ）。

A. $O(n^{3/2}m)$　　B. $O(nm)$　　C. $O(n^2m)$　　D. $O(nm^2)$

26.当输入为“20 2”时，输出的第一行为（ ）。

A. “4”　　B. “5”　　C. “6”　　D. “20”

27.（4 分）当输入为“100 100”时，输出的第一行为（ ）。

A. “6”　　B. “7”　　C. “8”　　D. “9”

（3）

```
01 #include <iostream>
02
03 using namespace std;
04
05 int n, k;
06
07 int solve1()
08 {
09
10      int l = 0, r = n;
11      whil  (l <= r) {
12          int mid = (l + r) / 2;
13          if (mid * mid <= n) l = mid + 1;
14          else r = mid - 1;
15    }
16    return l - 1;
17 }
18
19 double solve2(double x)
20 {
21    if (x == 0) return x;
22    for (int i = 0; i < k; i++)
23        x = (x + n / x) / 2;
24    return x;
25 }
26
27 int main()
```


公众号：黑猫编程
网址：https://noi.hioier.com

```
27 {
28     cin >> n >> k;
29     double ans = solve2(solve1());
30     cout << ans << ' ' << (ans * ans == n) << endl;
31     return 0;
32 }
```

假设 int 为 32 位有符号整数类型，输入的 n 是不超过 47000 的自然数、k 是不超过 int 表示范围的自然数，完成下面的判断题和单选题：

- 判断题

28. 该算法最准确的时间复杂度分析结果为$O(\log n + k)$。（ ）

29. 当输入为“9801 1”时，输出的第一个数为“99”。（ ）

30. 对于任意输入的 n，随着所输入 k 的增大，输出的第二个数会变成“1”。（ ）

31. 该程序有存在缺陷。当输入的 n 过大时，第 12 行的乘法有可能溢出，因此应当将 mid 强制转换为 64 位整数再计算。（ ）

- 单选题

32. 当输入为“2 1”时，输出的第一个数最接近（ ）。

A. 1　　B. 1.414　　C. 1.5　　D. 2

33. 当输入为“3 10”时，输出的第一个数最接近（ ）。

A. 1.7　　B. 1.732　　C. 1.75　　D. 2

34. 当输入为“256 11”时，输出的第一个数（ ）。

A. 等于 16　　B. 接近但小于 16

C. 接近但大于 16　　D. 前三种情况都有可能

三、完善程序（单选题，每小题 3 分，共计 30 分）

（1）（枚举因数）从小到大打印正整数 n 的所有正因数。

试补全枚举程序。

```
01 #include <bits/stdc++.h>
02 using namespace std;
03
04 int main() {
05   int n;
06   cin >> n;
```


公众号：黑猫编程
网址：https://noi.hioier.com

```
07
08   vector<int> fac;
09   fac.reserve((int)ceil(sqrt(n)));
10
11   int i;
12   for (i = 1; i * i < n; ++i) {
13     if (①) {
14       fac.push_back(i);
15     }
16   }
17
18   for (int k = 0; k < fac.size(); ++k) {
19     cout << ② << " ";
20   }
21   if (③) {
22     cout << ④ << " ";
23   }
24   for (int k = fac.size() - 1; k >= 0; --k) {
25     cout << ⑤ << " ";
26   }
27 }
```

35.①处应填（ ）

A. n % i == 0　　B. n % i == 1

C. n % (i-1) == 0　　D. n % (i-1) == 1

36.②处应填（ ）

A. n / fac[k]　　B. fac[k]

C. fac[k]-1　　D. n / (fac[k]-1)

37.③处应填（ ）

A. (i-1) * (i-1) == n　　B. (i-1) * i == n

C. i * i == n　　D. i * (i-1) == n

38.④处应填（ ）

A. n-i　　B. n-i+1

C. i-1　　D. I

39.⑤处应填（ ）

A. n / fac[k]　　B. fac[k]

C. fac[k]-1　　D. n / (fac[k]-1)

（2）（洪水填充）现有用字符标记像素颜色的 8x8 图像。颜色填充的操作描述如下：给定起始像素的位置和待填充的颜色，将起始像素和所有可达的像素（可达的定义：经过

公众号：黑猫编程
网址：https://noi.hioier.com

一次或多次的向上、下、左、右四个方向移动所能到达且终点和路径上所有像素的颜色都与起始像素颜色相同），替换为给定的颜色。

试补全程序。

```
01 #include <bits/stdc++.h>
02 using namespace std;
03
04 const int ROWS = 8;
05 const int COLS = 8;
06
07 struct Point {
08   int r, c;
09   Point(int r, int c) : r(r), c(c) {}
10 };
11
12 bool is_valid(char image[ROWS][COLS], Point pt,
13                int prev_color, int new_color) {
14   int r = pt.r;
15   int c = pt.c;
16   return (0 <= r && r < ROWS && 0 <= c && c < COLS &&
17           ① && image[r][c] != new_color);
18 }
19
20 void flood_fill(char image[ROWS][COLS], Point cur, int new_color) {
21   queue<Point> queue;
22   queue.push(cur);
23
24   int prev_color = image[cur.r][cur.c];
25   ②;
26
27   while (!queue.empty()) {
28     Point pt = queue.front();
29     queue.pop();
30
31     Point points[4] = {③, Point(pt.r - 1, pt.c),
32                        Point(pt.r, pt.c + 1), Point(pt.r, pt.c - 1)};
33     for (auto p : points) {
34       if (is_valid(image, p, prev_color, new_color)) {
35         ④;
36         ⑤;
37       }
38     }
39   }
```


公众号：黑猫编程
网址：https://noi.hioier.com

```
}

int main() {
  char image[ROWS][COLS] = {{'g', 'g', 'g', 'g', 'g', 'g', 'g', 'g'},
                            {'g', 'g', 'g', 'g', 'g', 'g', 'r', 'r'},
                            {'g', 'r', 'r', 'g', 'g', 'r', 'g', 'g'},
                            {'g', 'b', 'b', 'b', 'b', 'r', 'g', 'r'},
                            {'g', 'g', 'g', 'b', 'b', 'r', 'g', 'r'},
                            {'g', 'g', 'g', 'b', 'b', 'b', 'b', 'r'},
                            {'g', 'g', 'g', 'g', 'g', 'b', 'g', 'g'},
                           {'g', 'g', 'g', 'g', 'g', 'b', 'b', 'g'}};

  Point cur(4, 4);
  char new_color = 'y';

  flood_fill(image, cur, new_color);

  for (int r = 0; r < ROWS; r++) {
    for (int c = 0; c < COLS; c++) {
      cout << image[r][c] << " ";
    }
    cout << endl;
  }
  // 输出:
  // g g g g g g g g
  // g g g g g g r r
  // g r r g g r g g
  // g y y y y r g r
  // g g g y y r g r
  // g g g y y y y r
  // g g g g g y g g
  // g g g g g y y g

  return 0;
}
```

40.①处应填（　）

A. `image[r][c] == prev_color`

B. `image[r][c] != prev_color`

C. `image[r][c] == new_color`

D. `image[r][c] != new_color`

41.②处应填（　）

A. `image[cur.r+1][cur.c] = new_color`

公众号：黑猫编程
网址：https://noi.hioier.com

B. image[cur.r][cur.c] = new_color
C. image[cur.r][cur.c+1] = new_color
D. image[cur.r][cur.c] = prev_color

42.③处应填（ ）

A. Point(pt.r, pt.c)　　B. Point(pt.r, pt.c+1)
C. Point(pt.r+1, pt.c)　　D. Point(pt.r+1, pt.c+1)

43.④处应填（ ）

A. prev_color = image[p.r][p.c]　　B. new_color = image[p.r][p.c]
C. image[p.r][p.c] = prev_color　　D. image[p.r][p.c] = new_color

44.⑤处应填（ ）

A. queue.push(p)
B. queue.push(pt)
C. queue.push(cur)
D. queue.push(Point(ROWS,COLS))

公众号：黑猫编程
网址：https://noi.hioier.com