2021 CCF 非专业级别软件能力认证第一轮

(CSP-J)入门级 C++语言试题 模拟卷 – 9

考生注意事项:

1. 全部试题答案均要求写在答卷纸上，写在试卷纸上一律无效。

2. 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、单选题(共 15 题，每题 2 分，共计 30 分;每题有且仅有一一个正确答案)

1.操作系统是一 类重要的系统软件，下面几个软件不属于系统软件的是( )。

A. Mac OS　　B. Java　　C. Windows 10　　D. Unix

2. 在计算机内部，用来传送、存储、加工处理的数据或指令(命令)都是以()形式进行的。

A.十进制码　　B.二进制码　　C.智能拼音码　　D.五笔字型码

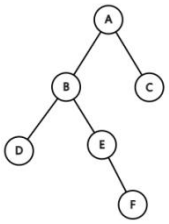
3. 右图是一棵二叉树，它的后序遍历是()。

A. ABDEFC

B. DBEFAC

C. DFEBCA

D. ABCDEF



4.寄存器是( )的重要组成部分。

A.硬盘　　B.高速缓存

C.内存　　D.中央处理器

5.在编程时，如果需要从磁盘文件中输入一个很大的二维数组（例如 1000*1000 的 double 型数组），按行读（即外层循环是关于行的）与按列读（即外层循环是关于列的）相比，在输入效率上（ ）。

A. 没有区别　　B. 按行读的方式要高一些

C. 按列读的方式要高一些　　D. 取决于数组的存储方式。

6. 与十进制数 28.5625 相等的四进制数是（ ）。

A. 123.21　　B. 131.22　　C. 130.22　　D. 130.21

7. 应用快速排序的分治思想，可以实现一 个求第 K 大数的程序。假定不考虑极端的最坏情况，理论上可以实现的最低的算法时间复杂度为( )。

A.O(n2)　　B.O(nlogn)　　C.O(n)　　D.O(1)

8.下列 IP 地址中正确的是()。

A.202.300.12.4　　B.192.168.0.3

C.100:128:35:91　　D.19.256.0.1

9.二进制数 10101110 和 11110110 做逻辑异或运算的结果是。

A.11111111　　B.10100110　　C.01011000　　D.01011010

10. 一棵包含 n 个结点的树有( ) 条边。

A.n−1　　B.n　　C.n−2　　D.无法确定

11.如果根结点的深度记为 1，则一棵恰有 2011 个叶子结点的二叉树的深度不可能是( )。

A.11　　B.12　　C.13　　D.2011

12 要求以下程序的功能是计算：s = 1 + 1/2 + 1/3 + ... + 1/10。

```
#include <iostream>
using namespace std;
int main() {
    int n;
    float s;
    s = 1.0;
    for (n = 10; n > 1; n--)
        s = s + 1 / n;
    cout << s << endl;
    return 0;
}
```

程序运行后输出结果错误，导致错误结果的程序行是（ ）。

A. s = 1.0;　　B. for (n = 10; n > 1; n--)

C. s = s + 1 / n;　　D. cout << s << endl;

13. 已知 6 个结点的二叉树的先根遍历是 1 2 3 4 5 6（数字为结点的编号，以下同），后根遍历是 3 2 5 6 4 1，则该二叉树的可能的中根遍历是（ ）

A. 3 2 1 4 6 5　　B. 3 2 1 5 4 6

C. 2 1 3 5 4 6　　D. 2 3 1 4 6 5

14.对于序列“7,5, 1,9, 3,6, 8,4”，在不改变顺序的情况下，去掉() 会使逆序对的个数减少 3。

A.7　　B.5　　C.4　　D.6

15.定义一一种字符串操作，一次可以将其中一个元素移到任意位置。举例说明，对于字符串“BCA”可以将 A 移到 B 之前，变字符串“ABC”。如果要将字符串“DACHEBGIF”变成“ABCDEFGHI" 最少需要( )次操作。

A.4　　B.5　　C.6　　D.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 4 分，共计 40 分）

```
1   #include <bits/stdc++.h>
2   using namespace std;
3   int main()
4   {
5       int a[4], b[4];
6       int i, j, tmp;
7       for (i = 0; i < 4; i++)
8           scanf("%d", &b[i]);
9       for (i = 0; i < 4; i++)
10      {
11          a[i] = 0;
12          for (j = 0; j <= i; j++)
13          {
14              a[i] += b[j];
15              b[a[i] % 4] += a[j];
16          }
17          tmp = 1;
18      }
19      for (i = 0; i < 4; i++)
20      {
21          a[i] %= 10;
22          b[i] %= 10;
23          tmp *= a[i] + b[i];
24      }
25      printf("%d\n", tmp);
```

```
26      return 0;
27  }
```

判断题

1)若输入的 b 数组都是奇数，程序运行到第 15 行时，b[2]没有发生变化。（ ）
2)若将第 14 行和第 15 行交换，程序的结果可能发生变化。（ ）
3)当程序运行到第 15 行时，此时 a 数组的值等于 b 数组的值。（ ）
4)若输入 b 数组都是偶数，当程序运行到第 15 行时，a 数组中的值也全部为偶数。（ ）

选择题

5)该程序能输出的最大结果为( )。

A.6561　　B.10000　　C.104976　　D.43046721

6)若输入 2 3 5 7，输出的结果为()。

A.210　　B.5850　　C.3360　　D.103680

```
1   #include <bits/stdc++.h>
2   #define maxn 50
3   using namespace std;
4   const int y = 2009;
5   int c[maxn][maxn];
6   int main() {
7       int n, i, j, s;
8       s = 0;
9       scanf("%d", &n);
10      c[0][0] = 1;
11      for (i = 1; i <= n; i++) {
12          c[i][0] = 1;
13          for (j = 1; j < i; j++)
14              c[i][j] = c[i - 1][j - 1] + c[i - 1][j];
15          c[i][i] = 1;
16      }
17      for (i = 0; i <= n; i++)
18          s = (s + c[n][i]) % y;
19      printf("%d\n", s);
20  }
```

判断题

1)若输入的 n 足够大时，对于任意的 i> j - 1，有 c[i][j]≥c[i][j-1]。（ ）
2)将第 13 行的“j=1” 改为“j=0”， 程序会出错。（ ）
3)对于任意 c[i][j](1<j≤i≤n),都有 c[i][j]>c[i−1][j]。（ ）
4)若给出 x(x>3),则任意的 c[x][i](1≤i≤x-1)和 x 的最大公因数一定大于 1。（ ）

选择题

5)若 c 数组中含有 x 个 1 时，n 是( )。

A. (x+1)/2　　B.$\log_2 x$　　C. (x+1)/2　　D. x

6)若输人 17，则输出的值为( )。

A. 487　　B.289　　C.153　　D.2008

3.

```
#include <stdio.h>
int main() {
    int n, m, i, j, k, p;
    int a[100], b[100];
    scanf("%d%d", &n, &m);
    a[0] = n;
    i = 0;
    p = 0;
    k = 1;
    do {
        for (j = 0; j < i; j++)
            if (a[i] == a[j]) {
                p = 1;
                k = j;
                break;
            }
        if (p)
            break;
        b[i] = a[i] / m;
        a[i + 1] = a[i] % m * 10;
        i++;
    } while (a[i] != 0);
    printf("%d.", b[0]);
    for (j = 1; j < k; j++)
        printf("%d", b[j]);
    if (p)
        printf("(");
    for (j = k; j < i; j++)
        printf("%d", b[j]);
    if (p)
        printf(")");
```

```
    printf("\n");
}
```

判断题

1)有可能找到一对 n 和 m 使得程序在第 10 行的 do-while 语句中形成死循环。(　)

2)程序不会输出小括号的充要条件是 n 和 m 的最大公因数不为 1。(　)

选择题

3)(2 分)当 n 和 m 均小于等于 10 且为正整数，要使输出的括号内只有一位数，则有( )种方案。

A.9　　B.18　　C.23　　D.42

4)(2 分)当 n=5, m=8 时，第 12 行比较执行次数是( )次。

A.5　　B.6　　C.7　　D.8

5)(2 分)当 0≤n≤30, m=n+1 时，使得程序输出小括号有( ) 种方案。

A.16　　B.18　　C.20　　D.22

6)(3 分)当 n=5, m=13 时，输出的小括号中的数字有( ) 位。

A.4　　B.5　　C.6　　D.7

三、完善程序(单选题，每小题 3 分，共计 30 分)

1. (最大连续子段和)给出一一个数列(元素个数不多于 100)， 数列元素均为负整数、正整数、0。请找出数列中的一个连续子数列，使得这个子数列中包含的所有元素之和最大，在和最大的前提下还要求该子数列包含的元素个数最多，并输出这个最大和以及该连续子数列中元素的个数。例如数列为 4，−5, 3, 2, 4 时，输出 9 和 3;数列为 1,2,3,−5,0, 7, 8 时，输出 16 和 7。

```
1  #include <iostream>
2  using namespace std;
3
4  int a[101];
5  int n, i, ans, len, tmp, beg, end;
6  int main() {
7      cin >> n;
8      for (i = 1; i <= n; i++)
9          cin >> a[i];
10     tmp = 0;
11     ans = 0;
12     len = 0;
13     beg =①;
14     for (i = 1; i <= n; i++) {
15         if (tmp + a[i] > ans) {
16             ans = tmp + a[i];
17             len = i − beg;
18         } else if (②&&i − beg > len)
```

```
19              len = i – beg;
20          if (tmp + a[i]③){
21              beg = ④;
22              tmp = 0;
23          }
24          else
25              ⑤;
26      }
27      cout << ans << "" << len << endl;
28      return 0;
29  }
```

1)①处应填( )。

A.0　　B.1　　C.n　　D.-1

2)②处应填( )。

A.tmp==ans　　C.a[i]+tmp==ans

B.a[j]+tmp<=ans　　D.tmp<=ans

3)③处应填( )。

A.==0　　B.<ans　　C.<0　　D.<=0

4)④处应填()。

A.1　　B.n　　C.i　　D.0

5)⑤处应填()。

A.tmp=0　　C.beg=i

B.len++　　D.tmp+=a[i]

2. (寻找等差数列)有一些长度相等的等差数列(数列中每个数都为 0~59 的整数)，设长度均为 L，将等差数列中的所有数打乱顺序放在一起。 现在给你这些打乱后的数，问原先 L 最大可能为多大?先读入一个数 n(1≤n≤60)，再读入 n 个数，代表打乱后的数。输出等差数列最大可能长度 L。

试补全程序。

```
1   #include <iostream>
2   using namespace std;
3
4   int hash[60];
5   int n, X, ans, maxnum;
6
7   int work(int now)
8   {
9       int first, second, delta, i;
10      int ok;
11      while (① && !hash[now])
12          ++now;
13      if (now > maxnum)
```

```
14              return 1
15          first = now;
16          for (second = frst; second <= maxnum; second++)
17              if (hash[second])
18              {
19                  delta = ②;
20                  if (first + delta * ③ > maxnum)
21                      break;
22                  if (delta == 0)
23                      ok = (④);
24                  else
25                  {
26                      ok = 1;
27                      for (i = 0; i < ans; i++)
28                          ok = ⑤ && (hash[first + delta * i]);
29                  }
30                  if (ok)
31                  {
32                      for (i = 0; i < ans; i++)
33                          hash[first + delta * i]--;
34                      if (work(first))
35                          return 1;
36                      for (i = 0; i < ans; i++)
37                          hash[first + delta * i]++;
38                  }
39              }
40          return 0
41  }
42
43  int main()
44  {
45          int i;
46          memset(hash, 0, sizeof(hash));
47          cin >> n;
48          maxnum = 0
49          for (i = 0; i < n; i++)
```

```
50      {
51          cin >> x;
52          hash[x]++;
53          if (x > maxnum)
54              maxnum = x;
55      }
56      for (ans = n; ans >= 1; ans--)
57          if (n % ans == 0 && work(0))
58          {
59              cout << ans << endl;
60              break;
61          }
62      return 0;
63  }
```

1)①处应填()。

A. now　　B. now<=maxnum　　C. now<=n　　D. hash[now]==0

2)②处应填( )。

A. maxnum-first　　B. second-first

C. maxnum-second　　D. hash[second] -hash[first]

3)③处应填( )。

A. ans　　B. (ans-l)　　C. now　　D. (now- 1)

4)④处应填()。

A. hash[second]>=ans　　B. ok

C. first+delta*(ans- 1)　　D. hash[first]==ans

5)⑤处应填()。

A. ok　　B. hash[first]

C. (hash[second+delta*i])　　D. hash[second]