2019CCF 非专业级别软件能力认证第一轮

（CSP-J）入门级 C++语言试题 A 卷

认证时间：2019 年 10 月 19 日 14:30~16:30

考生注意事项：

- 试题纸共有 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. 中国的国家顶级域名是（ ）

   A. .cn　　B. .ch　　C. .chn　　D. .china

2. 二进制数 11 1011 1001 0111 和 01 0110 1110 1011 进行逻辑与运算的结果是（ ）。

   A. 01 0010 1000 1011　　B. 01 0010 1001 0011

   C. 01 0010 1000 0001　　D. 01 0010 1000 0011

3. 一个 32 位整型变量占用（ ）个字节。

   A. 32　　B. 128　　C. 4　　D. 8

4. 若有如下程序段，其中 s、a、b、c 均已定义为整型变量，且 a、c 均已赋值（c 大于 0）

```
s = a;
for (b = 1; b <= c; b++) s = s - 1;
```

   则与上述程序段功能等价的赋值语句是（ ）

   A. s = a - c;　　B. s = a - b;　　C. s = s - c;　　D. s = b - c;

5. 设有 100 个已排好序的数据元素，采用折半查找时，最大比较次数为（ ）

   A. 7　　B. 10　　C. 6　　D. 8

6. 链表不具有的特点是（ ）

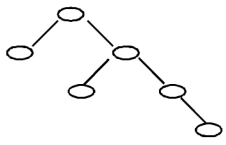
   A. 插入删除不需要移动元素　　B. 不必事先估计存储空间

   C. 所需空间与线性表长度成正比　　D. 可随机访问任一元素

7. 把 8 个同样的球放在 5 个同样的袋子里，允许有的袋子空着不放，问共有多少种不同的分法？（ ）提示：如果 8 个球都放在一个袋子里，无论是哪个袋子，都只算同一种分法

   A. 22　　B. 24　　C. 18　　D. 20

8. 一棵二叉树如右图所示，若采用顺序存储结构，即用一维数组元素存储该二叉树中的结点（根结点的下标为 1，若某结点的下标为 i ，则其左孩子位于下标 2i 处、右孩子位于下标 2i+1 处），则该数组的最大下标至少为（ ）。



A. 6　　B. 10　　C. 15　　D. 12

9. 100 以内最大的素数是（ ）。

A. 89　　B. 97　　C. 91　　D. 93

10. 319 和 377 的最大公约数是（ ）。

A. 27　　B. 33　　C. 29　　D. 31

11. 新学期开学了，小胖想减肥，健身教练给小胖制定了两个训练方案。方案一：每次连续跑 3 公里可以消耗 300 千卡（耗时半小时）；方案二：每次连续跑 5 公里可以消耗 600 千卡（耗时 1 小时）。小胖每周周一到周四能抽出半小时跑步，周五到周日能抽出一小时跑步。另外，教练建议小胖每周最多跑 21 公里，否则会损伤膝盖。请问如果小胖想严格执行教练的训练方案，并且不想损伤膝盖，每周最多通过跑步消耗多少千卡？（ ）

A. 3000　　B. 2500　　C. 2400　　D. 2520

12. 一副纸牌除掉大小王有 52 张牌，四种花色，每种花色 13 张。假设从这 52 张牌中随机抽取 13 张纸牌，则至少（ ）张牌的花色一致。

A. 4　　B. 2　　C. 3　　D. 5

13. 一些数字可以颠倒过来看，例如 0、1、8 颠倒过来还是本身，6 颠倒过来是 9，9 颠倒过来看还是 6，其他数字颠倒过来都不构成数字。类似的，一些多位数也可以颠倒过来看，比如 106 颠倒过来是 901。假设某个城市的车牌只由 5 位数字组成，每一位都可以取 0 到 9。请问这个城市最多有多少个车牌倒过来恰好还是原来的车牌？（ ）

A. 60　　B. 125　　C. 75　　D. 100

14. 假设一棵二叉树的后序遍历序列为 DGJHEBIFCA，中序遍历序列为 DBGEHJACIF，则其前序遍历序列为（ ）。

A. ABCDEFGHIJ　　B. ABDEGHJCFI　　C. ABDEGJHCFI　　D. ABDEGHJFIC

15. 以下哪个奖项是计算机科学领域的最高奖？（ ）

A. 图灵奖　　B. 鲁班奖　　C. 诺贝尔奖　　D. 普利策奖

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.

```
1  #include <cstdio>
2  #include <cstring>
3  using namespace std;
4  char st[100];
5  int main() {
6    scanf("%s", st);
7    int n = strlen(st);
8    for (int i = 1; i <= n; ++i) {
9      if (n % i == 0) {
10       char c = st[i - 1];
11       if (c >= 'a')
12         st[i - 1] = c - 'a' + 'A';
13     }
14   }
15   printf("%s", st);
16   return 0;
17 }
```

- 判断题

1) 输入的字符串只能由小写字母或大写字母组成。（　）

2) 若将第 8 行的“`i = 1`”改为“`i = 0`”，程序运行时会发生错误。（　）

3) 若将第 8 行的“`i <= n`”改为“`i * i <= n`”，程序运行结果不会改变。（　）

4) 若输入的字符串全部由大写字母组成，那么输出的字符串就跟输入的字符串一样。（　）

- 选择题

5) 若输入的字符串长度为 18，那么输入的字符串跟输出的字符串相比，至多有（　）个字符不同。

A. 18　　B. 6　　C. 10　　D. 1

6) 若输入的字符串长度为（　），那么输入的字符串跟输出的字符串相比，至多有 36 个字符不同。

A. 36　　B. 100000　　C. 1　　D. 128

2.

```
1  #include <cstdio>
2  using namespace std;
3  int n, m;
4  int a[100], b[100];
5
6  int main() {
7    scanf("%d%d", &n, &m);
8    for (int i = 1; i <= n; ++i)
9      a[i] = b[i] = 0;
10   for (int i = 1; i <= m; ++i) {
11     int x, y;
12     scanf("%d%d", &x, &y);
13     if (a[x] < y && b[y] < x) {
14       if (a[x] > 0)
15         b[a[x]] = 0;
16       if (b[y] > 0)
17         a[b[y]] = 0;
18       a[x] = y;
19       b[y] = x;
20     }
21   }
22   int ans = 0;
23   for (int i = 1; i <= n; ++i) {
24     if (a[i] == 0)
25       ++ans;
26     if (b[i] == 0)
27       ++ans;
28   }
29   printf("%d\n", ans);
30   return 0;
31 }
```

假设输入的 **n** 和 **m** 都是正整数，**x** 和 **y** 都是在**[1, n]**的范围内的整数，完成下面的判断题和单选题：

- 判断题

1) 当 m>0 时，输出的值一定小于 2n。（　）

2) 执行完第 27 行的“++ans”时，ans 一定是偶数。（　）

3) a[i]和 b[i]不可能同时大于 0。（　）

4) 若程序执行到第 13 行时，x 总是小于 y，那么第 15 行不会被执行。（ ）

- 选择题

5) 若 m 个 x 两两不同，且 m 个 y 两两不同，则输出的值为（ ）

A. 2n-2m　　B. 2n+2　　C. 2n-2　　D. 2n

6) 若 m 个 x 两两不同，且 m 个 y 都相等，则输出的值为（ ）

A. 2n-2　　B. 2n　　C. 2m　　D. 2n-2m

3.

```
1  #include <iostream>
2  using namespace std;
3  const int maxn = 10000;
4  int n;
5  int a[maxn];
6  int b[maxn];
7  int f(int l, int r, int depth) {
8    if (l > r)
9      return 0;
10   int min = maxn, mink;
11   for (int i = l; i <= r; ++i) {
12     if (min > a[i]) {
13       min = a[i];
14       mink = i;
15     }
16   }
17   int lres = f(l, mink - 1, depth + 1);
18   int rres = f(mink + 1, r, depth + 1);
19   return lres + rres + depth * b[mink];
20 }
21 int main() {
22   cin >> n;
23   for (int i = 0; i < n; ++i)
24     cin >> a[i];
25   for (int i = 0; i < n; ++i)
26     cin >> b[i];
27   cout << f(0, n - 1, 1) << endl;
28   return 0;
29 }
```

- 判断题

1) 如果 a 数组有重复的数字，则程序运行时会发生错误。（ ）

2) 如果 b 数组全为 0，则输出为 0。（ ）

- 选择题

3) 当 n=100 时，最坏情况下，与第 12 行的比较运算执行的次数最接近的是：（ ）。

A. 5000　　B. 600　　C. 6　　D. 100

4) 当 n=100 时，最好情况下，与第 12 行的比较运算执行的次数最接近的是：（ ）。

A. 100　　B. 6　　C. 5000　　D. 600

5) 当 n=10 时，若 b 数组满足，对任意$0 \le i < n$，都有$b[i] = i + 1$，那么输出最大为（ ）。

A. 386　　B. 383　　C. 384　　D. 385

6) （4 分）当 n=100 时，若 b 数组满足，对任意$0 \le i < n$，都有$b[i] = 1$，那么输出最小为（ ）。

A. 582　　B. 580　　C. 579　　D. 581

三、完善程序（单选题，每小题 3 分，共计 30 分）

1. （矩阵变幻）有一个奇幻的矩阵，在不停的变幻，其变幻方式为：数字0变成矩阵$\begin{bmatrix}0 & 0\\0 & 1\end{bmatrix}$，数字1变成矩阵$\begin{bmatrix}1 & 1\\1 & 0\end{bmatrix}$。最初该矩阵只有一个元素0，变幻 n 次后，矩阵会变成什么样？

例如，矩阵最初为：[0]；矩阵变幻 1 次后：$\begin{bmatrix}0 & 0\\0 & 1\end{bmatrix}$；矩阵变幻 2 次后：

$$\begin{bmatrix}0 & 0 & 0 & 0\\0 & 1 & 0 & 1\\0 & 0 & 1 & 1\\0 & 1 & 1 & 0\end{bmatrix}。$$

输入一行一个不超过 10 的正整数 n。输出变幻 n 次后的矩阵。

试补全程序。

提示：

“<<”表示二进制左移运算符，例如$(11)_2 << 2 = (1100)_2$；

而“^”表示二进制异或运算符，它将两个参与运算的数中的每个对应的二进制位一一进行比较，若两个二进制位相同，则运算结果的对应二进制位为 0，反之为 1。

```
1  #include <cstdio>
2  using namespace std;
3  int n;
4  const int max_size = 1 << 10;
5
6  int res[max_size][max_size];
7
8  void recursive(int x, int y, int n, int t) {
9    if (n == 0) {
10     res[x][y] = ①;
11     return;
12   }
13   int step = 1 << (n - 1);
14   recursive(②, n - 1, t);
15   recursive(x, y + step, n - 1, t);
16   recursive(x + step, y, n - 1, t);
17   recursive(③, n - 1, !t);
18 }
19
20 int main() {
21   scanf("%d", &n);
22   recursive(0, 0, ④);
23   int size = ⑤;
24   for (int i = 0; i < size; ++i) {
25     for (int j = 0; j < size; ++j)
26       printf("%d", res[i][j]);
27     puts("");
28   }
29   return 0;
30 }
```

1) ①处应填（ ）

A. `n % 2`　　B. `0`　　C. `t`　　D. `1`

2) ②处应填（ ）

A. `x - step, y - step`　　B. `x, y - step`

C. `x - step, y`　　D. `x, y`

3) ③处应填（ ）

A. `x - step, y - step`　　B. `x + step, y + step`

C. `x - step, y`　　D. `x, y - step`

4) ④处应填（ ）

A. `n - 1, n % 2`　　B. `n, 0`
C. `n, n % 2`　　D. `n - 1, 0`

5) ⑤处应填（　）
A. `1 << (n + 1)`　　B. `1 << n`
C. `n + 1`　　D. `1 << (n - 1)`

2. （计数排序）计数排序是一个广泛使用的排序方法。下面的程序使用双关键字计数排序，将 n 对 10000 以内的整数，从小到大排序。

例如有三对整数（3,4）、（2,4）、（3,3），那么排序之后应该是（2,4）、（3,3）、（3,4）。
输入第一行为 n，接下来 n 行，第 i 行有两个数 a[i]和 b[i]，分别表示第 i 对整数的第一关键字和第二关键字。
从小到大排序后输出。

数据范围 $1 \le n \le 10^7, 1 \le a[i], b[i] \le 10^4$。

提示：应先对第二关键字排序，再对第一关键字排序。数组 ord[]存储第二关键字排序的结果，数组 res[]存储双关键字排序的结果。
试补全程序。

```
1  #include <cstdio>
2  #include <cstring>
3  using namespace std;
4  const int maxn = 10000000;
5  const int maxs = 10000;
6
7  int n;
8  unsigned a[maxn], b[maxn], res[maxn], ord[maxn];
9  unsigned cnt[maxs + 1];
10
11 int main() {
12   scanf("%d", &n);
13   for (int i = 0; i < n; ++i)
14     scanf("%d%d", &a[i], &b[i]);
15   memset(cnt, 0, sizeof(cnt));
16   for (int i = 0; i < n; ++i)
17     ①;   // 利用 cnt 数组统计数量
18   for (int i = 0; i < maxs; ++i)
19     cnt[i + 1] += cnt[i];
20   for (int i = 0; i < n; ++i)
21     ②;   // 记录初步排序结果
```

```
22   memset(cnt, 0, sizeof(cnt));
23   for (int i = 0; i < n; ++i)
24     ③;   // 利用 cnt 数组统计数量
25   for (int i = 0; i < maxs; ++i)
26     cnt[i + 1] += cnt[i];
27   for (int i = n - 1; i >= 0; --i)
28     ④;   // 记录最终排序结果
29   for (int i = 0; i < n; ++i)
30     printf("%d %d\n", ⑤);
31   return 0;
32 }
```

1) ①处应填（　）

A. ++cnt[i]
B. ++cnt[b[i]]
C. ++cnt[a[i] * maxs + b[i]]
D. ++cnt[a[i]]

2) ②处应填（　）

A. ord[--cnt[a[i]]] = i
B. ord[--cnt[b[i]]] = a[i]
C. ord[--cnt[a[i]]] = b[i]
D. ord[--cnt[b[i]]] = i

3) ③处应填（　）

A. ++cnt[b[i]]
B. ++cnt[a[i] * maxs + b[i]]
C. ++cnt[a[i]]
D. ++cnt[i]

4) ④处应填（　）

A. res[--cnt[a[ord[i]]]] = ord[i]
B. res[--cnt[b[ord[i]]]] = ord[i]
C. res[--cnt[b[i]]] = ord[i]
D. res[--cnt[a[i]]] = ord[i]

5) ⑤处应填（　）

A. a[i], b[i]
B. a[res[i]], b[res[i]]
C. a[ord[res[i]]], b[ord[res[i]]]
D. a[res[ord[i]]], b[res[ord[i]]]

2019CCF 非专业级别软件能力认证第一轮

（CSP-J）入门级参考答案

一、单项选择题（共 15 题，每题 2 分，共计 30 分）

1	2	3	4	5	6	7	8	9	10
A	D	C	A	A	D	C	C	B	C
11	12	13	14	15					
C	A	C	B	A					

二、阅读程序（除特殊说明外，判断题 1.5 分，单选题 3 分，共计 40 分）

第 1 题	判断题（填√或×）				单选题	
	1)	2)	3)	4)	5)	6)
	×	√	×	√	B	B

第 2 题	判断题（填√或×）				单选题	
	1)	2)	3)	4)	5)	6)
	√	×	×	×	A	A

第 3 题	判断题（填√或×）		单选题			
	1)	2)	3)	4)	5)	6)（4 分）
	×	√	A	D	D	B

三、完善程序（单选题，每小题 3 分，共计 30 分）

第 1 题					第 2 题				
1)	2)	3)	4)	5)	1)	2)	3)	4)	5)
C	D	B	B	B	B	D	C	A	B

青藤编程营 2019 CSP-J 第一轮 答案解析

一：

1. `.cn`：中国的顶级域名是 .cn，概念性知识，但大家注册的时候其实对这个后缀非常熟悉了，rg.noi.cn
2. `01 0010 1000 0011`：与运算是有 0 就为 0，都是 1 才是 1，所以比较快速的做法是看有没有 0。
3. `4`：1 Byte = 8 bits
4. `s=a-c`：过去考过类似的，既然是做了 c 次 -1，自然等价于 -c。
5. `7`：1(50),2(25),3(13),4(7),5(4),6(2),7(1)
6. `可随意访问任一元素`：链表要访问到某个元素，必须一个个找过去。
7. `18`：我们保证放法升序就能没有遗漏枚举了，实际考场中采用此方法是最稳妥的，可以避免推错。8，17，26，35，44，116，125，134，224，233，1115....当然从这里我们也能发现递推方程。
8. `15`：非常良心的考了原题，跟坐标是1，往右依次就是 3、7、15
9. `97`：这个就没啥说的了。。
10. `29`：如果辗转相除啥的可能会比较麻烦，最简单的方式就是验证一下四个选项是否除得尽。
11. `2400`：阅读理解题，做法就按照说法来跑就好。
12. `4`：抽屉原理，最坏情况就是最平均的情况，123412341234，还有一张必然会构成4。
13. `75`：xyzyx 模式，其中 z 必须是(0,1,8)中的一个,y必须是(0,1,8,6,9)中的一个，前面是 6 后面就是 9，x 同 y，所以乘法原理后就是，`3*5*5=75`
14. `ABDEGHJCFI`：经典的内容，在后序遍历找根然后在中序遍历划分左右子树即可。
15. `图林奖`：果然有你

二 - (1)

1. `x`：输入的字符并没有什么限制。
2. `√`：我们下面索引用到了 i-1
3. `x`：在 i>sqrt(n) 时也是存在因子的
4. `√`：这个程序在做的就是特定位置小写转大写。
5. `6`：只有 18 的因子的位置会被判断是否需要转换，共 6 个
6. `100000`：验证每个选项，发现另外三个因子数明显没有36

二 - (2)

非常绕的一道题，也是让大家叫苦连天的题。到底在做什么呢？很快能发现 a 与 b 数组在做的是建立 x~y 的链接，设置的时候也是对称设置的。

如果 x 原来对应的 a[x] 比新的 y 小，并且 y 原来对应的 b[y] 比新的 x 小，那么把原来的对应关系清空，建立新的关系。

1. `√`：只要有关系来，不管会不会覆盖，至少不会关系全空。
2. `x`：看上去是堆成的，但是因为有可能 x=1 对应着 y=2,那么做到 i==1 时 a[1]==2,b[1]==0;
3. `x`：如果我们建立了 2~3 与 3~2 两组关系明显就可以了。
4. `x`：15 行是在清楚原来 x 所对应的关系，和题目表述没啥联系。
5. `2n-2m`：此时不会发生覆盖的情况，每组链接都被做了。
6. `2n-2`：最后只会保留一组关系

二 - (3)

经验丰富的同学会看出来，这里就是在建立一颗树，每次找到当前区间最小值的位置作为根，然后划分左右做下一层。

最后的返回值非常重要，返回的是左边的值+右边的值+深度*当前根权值，其实就是根节点权值为1，后续按照深度定全职

1. `x`：题目中找最小值有多个相等的话是随便选一个，不会有问题的
2. `√`：b数组是每个点的 value，a 数组是每个点的 key，最后计算是计算value*depth，所以b数组全 0 结果就是全 0
3. `5000`：每一层都要做一次，最坏情况下是一开始有序，每次都是第一个是根，所以要做 100 层，第一次 100 次，每层次数 -1，1+2+3+...+100 = 5050
4. `600`：最好情况下就是每次都二分，层数应该是 log2(100)，也就是6左右。此时每层都差不多100个（后面随着有的点做了根会略少）。
5. `385`：最坏情况就是`1*1+2*2+3*3....`，手算一下，很快发现选择385.
6. `580`：最好情况就是刚好每次都二分，那么 100 个节点的每层节点数量就是 （1,2,4,8,16,32,37）分别乘对应权值即可 1+4+12+32+80+192+259

三 -（1）

这题很明显是利用递归填好每个的位置，从最大的开始递归往下做。

1. `t`：x，y指当前的矩阵左上角，n是多少阶，那么t自然就是当前是 0 还是 1 了。
2. `x,y`：左上角矩阵
3. `x+step,y+step`：第二个是右上，第三个是左下，那么剩下的自然是右下了。
4. `n,0`：一开始的时候是 n 阶，0 为特征做下去。
5. `1<<n`：最终大小自然是 2 的 n 次幂，这边复合的就是位运算的方式的结果。

三 -（2）

题目其实提示的非常充分了，空还是很好填的

1. `++cnt[b[i]]`：题目说了，先按照b排序，那么自然是用b数组做关键字。
2. `ord[--cnt[b[i]]] = i`：理解这里最重要的是搞懂 18~19 行的内容， 18~19 行在处理我们上限范围内的前缀和，而其实本质上就是每个人的排名了。所以这边我们的`cnt[b[i]]`就是排名了，后续就很好理解了。
3. `++cnt[a[i]]`：现在做a，同样的方式即可。
4. `res[--cnt[a[ord[i]]]] = ord[i]`：同上，此时我们采用 res 数组来处理最终结果。
5. `a[res[i]],b[res[i]]`：输出最终顺序的 a 与 b。