2023 LGR 非专业级别软件能力认证第一轮

（SCP-S）提高级 C++语言模拟试题

认证时间：2023 年 8 月 14 日 14:30~16:30

考生注意事项：

- 试题纸共有 14 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. 以下物品可以携带进 CSP 第二轮测试考场的是（　　）。

A. 带有计算器功能的直尺
B. Um_nik 签名限量款背包
C. 印有完整 Splay 代码的文化衫 T 恤
D. 一大份绿鸟牌烤鸡柳

2. 在 CSP 第二轮测试中，老 A 发现小 K 正在使用某种技术手段（比如 SSH）抄袭自己的代码，若老 A 未及时举报小 K 的行为，且比赛结束后查出两人代码雷同，且有证据证明老 A 对此事知情，则老 A 与小 K 分别受到的惩罚禁赛时间为：（　　）。

A. 三年，三年　　B. 一年，三年
C. 一年，一年　　D. 无惩罚，三年

3. 「流程结构」是编程中用于控制程序执行流程的一种方式．它包括顺序结构、分支结构和循环结构．在一些诗歌作品中，也有对「流程结构」的体现．下列诗歌片段中体现循环结构的是（　　）。

A. 如果还能找到你，就让风儿告诉你。　　——《Artificial Emotions》
B. 只要我还能够行走，只要我还能够张望，只要我还能够呼吸，就一直走向前方。
——《Песня отревожной молодости》
C. 昔闻洞庭水，今上岳阳楼。
——《登岳阳楼》
D. 啊如果我在，战斗中牺牲，啊朋友再见吧，再见吧，再见吧！如果我在，战斗中牺牲，你一定把我来埋葬。
——《Bella Ciao》

4. 以下四种编程语言中，属于弱类型编程语言的是：

A. Java　　B. Go
C. Rust　　D. C++

5. 现有一段长度为 3 分钟，采样率为 44.1 千赫兹、位深为 16 比特的 wav 文件的双通道立体声音频，该音频文件占用的存储空间大小最接近：（　　）。

A. 15 MiB　　B. 30 MiB

C. 120 MiB　　D. 240 MiB

6. 在 vim 编辑器中，若当前位于 normal 模式，你需要将光标移动到这一行的末尾并进入 insert 模式．依次按下以下按键无法实现该目标的是：。

A. Esc、:、w、q、回车　　B. $、a

C. A　　D. o、退格

7. 在下列四组时空限制中，书写正确且允许使用的内存最多的一组是：（　　）

A. 10 s，256000 KiB

B. 4000 kg，10^{24} bit

C. 3000000 μs，1024 Mbit

D. 2000 ms，256 MiB

8. 假设某算法的计算时间表示为递推关系式 $T(1)=\Theta(1)$，$T(n)=T(\lceil\frac{\sqrt{2}}{2+\sqrt{2}}n\rceil)+\Theta(\lg n)$，则算法的时间复杂度为（　　）。

A. $\Theta(\lg n)$　　B. $\Theta(\sqrt{n}\lg n)$

C. $\Theta(\lg(\lg n))$　　D. $\Theta((\lg n)^2)$

9. 给定矩阵：$A=\begin{pmatrix}1 & 2\\ 3 & 4\end{pmatrix}$。我们定义一次操作为：选择同行或同列的两个元素，令其中一个加一，另一个减一。下列矩阵中不能通过 A 矩阵进行若干次操作得到的是（）。

A. $\begin{pmatrix}-3 & 6\\ 7 & 0\end{pmatrix}$　　B. $\begin{pmatrix}4 & -1\\ 0 & 7\end{pmatrix}$

C. $\begin{pmatrix}-1 & 4\\ 5 & 2\end{pmatrix}$　　D. $\begin{pmatrix}-1 & 5\\ 6 & 1\end{pmatrix}$

10. 「平衡三进制」是一种特殊的计数体系，与标准三进制相比，其中用于计数的符码为 -1, 0, 1，为书写方便，通常使用字母 Z 代替-1。例如，因为 $1\times3^2+(-1)\times3^1+0\times3^0=6$，所以十进制的 6 用平衡三进制表示为 1Z0．则十进制的 -11 可用平衡三进制表示为（　　）：

A. Z0Z　　B. ZZ0

C. Z1Z　　D. ZZ1

11. 竞赛图也叫有向完全图。每对顶点之间都有一条边相连的有向图称为竞赛图。5 个点的无标号竞赛图数是：（　　）。

A. 9　　B. 10

C. 11　　D. 12

12. 依次抛出四个六面骰子，按照抛出顺序将骰子上的数值记为 a,b,c,d。则 a < b; b > c; c < d 同时成立的概率为（　　）。

A. 95/648　　B. 4/27

C. 5/27　　D. 1/6

13. 有 6 堆石子，每堆石子分别有 1,4,7,1,5,4 个。Alice 和 Bob 两人轮流操作。轮到 Alice 时需要选择一堆石子并拿走其中的任意正奇数个，轮到 Bob 时需要选择一堆石子并拿走其中的任意正偶数个，最先无法操作的人判输。下列说法正确的是：（　　）

A. 若 Alice 先操作，则 Alice 有必胜策略；若 Bob 先操作，则 Alice 有必胜策略。

B. 若 Alice 先操作，则 Alice 有必胜策略；若 Bob 先操作，则 Bob 有必胜策略。

C. 若 Alice 先操作，则 Bob 有必胜策略；若 Bob 先操作，则 Alice 有必胜策略。

D. 若 Alice 先操作，则 Bob 有必胜策略；若 Bob 先操作，则 Bob 有必胜策略。

14. 阅读以下 C++代码片段：

```
uint32_t Trans(uint32_t x) {
  x ^= x << 13; x ^= x >> 17; x ^= x << 5;
  return x;
}
void Cycle(uint32_t x) {
  uint32_t old = x, cnt = 0;
  do {
    x = Trans(x); ++cnt;
  } while (x != old);
}
```

其中 uint32_t 表示无符号 32 位整数，在大多数环境中等同于 unsigned int。下列说法有误的一项是：（　）。

A. 若将任意 uint32_t 值代入 Trans 函数，得到的返回值一定和输入参数不同。

B. 若两 uint32_t 变量 x 与 y 的值不同，则 Trans(x)与 Trans(y)的值一定不同。

C. 将任意 uint32_t 值代入 Cycle 函数，则函数执行时++cnt;一行被执行到的次数一定为奇数。

D. 若将任意 uint32_t 值代入 Cycle 函数，一定不会出现无限循环。

15. 观察如下代码片段：

```
union U{
   bool flag1, flag2, flag3, flag4, flag5;
   signed   short a;
   unsigned short b;
   enum E{
       CardA = 0, CardB = 1,
       CardC = 2, CardD = 142857
   } e;
} u;
```

其中， sizeof(u) 的值为（　　）。

A. 4　　B. 8
C. 13　　D. 16

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 2 分，选择题 3 分，共计 40 分）

(1)

```
01 #include<iostream>
02 using namespace std;
03 int a[100005], b[100005], n, m;
04 void very_quick_sort(int l, int r, int p, int q){
05     if(l >= r || p > q){    // ①
06         return;
07     }
08     int mid = (l + r) / 2;
09     int p0 = p - 1;
10     int q0 = q + 1;
11     for(int i = p;i <= q;i ++){
12         if(a[i]  > mid) b[++ p0] = a[i];
13             else        b[-- q0] = a[i];
14     }
15     for(int i = p;i <= q;i ++)
16         a[i] = b[i];
17     very_quick_sort(mid + 1, r, p, p0);
18     very_quick_sort(l, mid, q0, q);
19 }
20 int main(){
21     cin >> n >> m;
22     for(int i = 1;i <= n;i ++)
23         cin >> a[i];
24     very_quick_sort(1, m, 1, n);
25         // ②
26     for(int i = 1;i <= n;i ++)
27         cout << a[i] << " ";
28     cout << endl;
29     return 0;
30 }
```

保证输入的 n 不超过 10^5，m 不超过 10^9，且 $1 \le a_1, a_2, \cdots, a_n \le m$。完成下面的判断题和单选题

- 判断题

16. (1 分)上述代码实现了一种排序算法，可以将 a 数组按照从小到大的顺序排序。（　　）

17. 如果在程序开始之前向 b 数组里写入数据（保证不会发生数组越界），则上述代码的输出不会发生变化。（　　）

18. 若 $n = m$，存在某种数据构造方式，使得上述代码运行的时间复杂度为 $O(n^2)$，这是因为算法本身是对快速排序的改进，但是这种改进不能避免由于对数组的划分不够均等而在极端数据下导致复杂度发生退化。（　　）

19. 如果将 ① 处的 `l >= r` 条件删除（同时删除 `||` 使得程序能正常编译运行，下同），程序的时间复杂度不会发生变化；而将 `p > q` 条件删除，程序在某些数据下的运行效率将会明显降低。（　　）

- 单选题

20. 不认为 n, m 同阶，即可能出现 n 远大于 m 或者 m 远大于 n 的情况。则该程序的最坏时间复杂度为（　　）。

A. $\Theta(n^2 + m^2)$　　B. $\Theta(\mathrm{m \log m})$

C. $\Theta(\mathrm{m \log n})$　　D. $\Theta(\mathrm{n \log m})$

21. 若输入数据为：

```
10 10
10 4 5 2 2 3 1 5 8 3
```

那么在程序执行到 ② 位置时， b 数组 内的值为 （

A. [10,8,3,5,1,3,2,2,5,4]

B. [3,5,1,3,2,2,5,4,10,8]

C. [10,8,5,5,4,3,3,2,2,1]

D. [1,2,2,3,3,4,5,5,8,10]

(2)

```
01 #include <iostream>
02 #include <vector>
03 #include <algorithm>
04 using namespace std;
05 const int mod = 1000000000 + 7;
06 int w0[100005];
07 int w1[100005];
08 int w2[100005];
09 int n, m, k, f[100005], d[100005], id[100005];
10 vector <int> e[100005];
11 void dfs(int u, int fa){
12     d[u] = d[fa] + 1;
13     f[u] = fa;
14
15     for(auto &v : e[u]) if(v != fa){
16         dfs(v, u);
17     }
```

```
}
bool cmp(int a, int b){
    return d[a] < d[b];
}
int main(){
    cin >> n >> m >> k;
    for(int i = 2;i <= n;i ++){
        int u, v;
        cin >> u >> v;
        e[u].push_back(v);
        e[v].push_back(u);
    }
    dfs(1, 0);                      // ①
    for(int i = 1;i <= m;i ++){
        int x, w;
        cin >> x >> w;
        w1[x] = (w1[x] + w) % mod;
        w2[x] = (w2[x] + w) % mod;
    }
    for(int i = 1;i <= n;i ++)
        id[i] = i;
    sort(id + 1, id + 1 + n, cmp);
    for(int i = 1;i <= k;i ++){
        for(int j = n;j >= 1;j --){
            int x = id[j];
            for(auto &y : e[x]) if(y != f[x]){
                w1[y] = (w1[y] + w1[x]) % mod;
            }
            w1[x] = 0;
        }
        for(int x = 1;x <= n;x ++)
            w1[x] = (w1[x] - w0[x] + mod) % mod,
            w0[x] = 0;
        for(int j = 1;j <= n;j ++){ // ②
            int x = id[j];
            if(f[x]){
                w1[f[x]] = (w1[f[x]] + w2[x]) % mod;
                w2[f[x]] = (w2[f[x]] + w2[x]) % mod;
                w0[x] = (w0[x] + w2[x]) % mod;
                w2[x] = 0;
            }
        }
    }
    for(int i = 1;i <= n;i ++)
```

```
62         cout << w1[i] << " ";
63     return 0;
64 }
```

保证输入的 n,m 不超过 10^5，k 不超过 20，且 $1\le x_i\le n$，$0\le w_i<10^9+7$。**完成下面的判断题和单选题：**

- 判断题

22. 如果更改 ① 处 dfs(1, 0) 为 dfs(n, 0)，则输出结果可能有变化。(　　)

23. （1 分）如果 k=n，那么输出结果均为 0。(　　)

24. 如果更改 ② 处 for(int j=1;j<=n;j++) 为 for(int j=n;j>=1;j--)，那么对于任意合法的输入数据，更改前后程序的输出均相同。(　　)

- 单选题

25. （2 分）该程序的时间复杂度为（　　）。

A. O(nmk)　　B. O(nk+m)
C. O(km+n)　　D. $O(n+m+2^k)$

26. 对于以下的输入数据，输出结果为（　　）。

```
5 2 1
1 2
2 3
3 4
3 5
1 5
3 2
```

A. 0 7 0 2 2　　B. 2 7 2 3 2
C. 5 2 1 1 1　　D. 0 2 1 1 2

27. 对于以下的输入数据，输出结果为（　　）。

```
9 9 2
1 2
1 7
2 3
2 4
7 8
4 5
4 6
8 9 （接续右栏）
```

```
1 1
2 10
3 100
4 1000
5 10000
6 100000
7 1000000
8 10000000
9 100000000
```

A. 10001100 1110000 1001 101 100010 10010 100000010 1 1000000
B. 0 0 1 1 10 10 0 1 1000000
C. 11001110 1111100 1001 110101 100010 10010 110000010 100000001 1000000
D. 11001111 1111112 1121 111121 112010 112010 111000012 112000001 121000000

(3)

```
01  #include<iostream>
02  using namespace std;
03  typedef long long i64;
04  namespace CirnoTree{
05      const int SIZ = 4e6 + 3;
06
07      int build1(int l, int r);
08      int build2(int l, int r);
09
10      int siz = 0;
11      int lft[SIZ], rgt[SIZ];
12      int nex[SIZ], son[SIZ];
13      i64 sum[SIZ];
14      i64 below_tag[SIZ];
15      i64 right_tag[SIZ];
16
17      int build1(int l, int r){
18          int u = ++ siz;
19          lft[u] = l;
20          rgt[u] = r;
21          son[u] = l == r ? 0 : build2(l, r);
22          return u;
23      }
24      int build2(int l, int r){
25          int mid = (l + r) / 2;
26          int p = build1(l, mid);
27          nex[p] = l == r ? 0 : build2(mid + 1, r);
28          return p;
29      }
30      void update(int t){
31          if(below_tag[t] != 0 && son[t] != 0){
32              sum[son[t]] +=
33                  1ll * (rgt[son[t]]-lft[son[t]]+1) * below_tag[t];
34              below_tag[son[t]] += below_tag[t];
35              right_tag[son[t]] += below_tag[t];
36          }
37          below_tag[t] = 0;
38          if(right_tag[t] != 0 && nex[t] != 0){
39              sum[nex[t]] +=
40                  1ll * (rgt[nex[t]]-lft[nex[t]]+1) * right_tag[t];
41              below_tag[nex[t]] += right_tag[t];
```

```
            right_tag[nex[t]] += right_tag[t];
        }
        right_tag[t] = 0;
    }
    void modify(int t, int pos, int w){
        if(rgt[t] <= pos){
            below_tag[t] += w;
            sum[t] += 1ll * w * (rgt[t] - lft[t] + 1);
        } else {
            int l = max(lft[t],   1);
            int r = min(rgt[t], pos);
            update(t);
            sum[t] += 1ll * (r - l + 1) * w;
            for(int p = son[t];p != 0 && lft[p] <= pos;p = nex[p])
                update(p), modify(p, pos, w);
        }
    }
    i64 query(int t, int pos){
        if(rgt[t] <= pos){
            return sum[t];
        } else {
            update(t); i64 ans = 0;
            for(int p = son[t];p != 0 && lft[p] <= pos;p = nex[p])
                update(p), ans += query(p, pos);
            return ans;
        }
    }
};
int main(){
    int n, m, root;
    cin >> n >> m;
    root = CirnoTree :: build1(1, n);
    for(int i = 1, x;i <= n;i ++){
        cin >> x;
        CirnoTree :: modify(root, i, x);
    }
    for(int i = 1, op;i <= m;i ++){
        cin >> op;
        if(op == 1){
            int l, r, w;
            cin >> l >> r >> w;
            CirnoTree :: modify(root, r, w);
            CirnoTree :: modify(root, l - 1, -w);
        } else {
```

```
86              int l, r;
87              cin >> l >> r;
88              i64 w1 = CirnoTree :: query(root, r);
89              i64 w2 = CirnoTree :: query(root, l - 1);
90              cout << w1 - w2 << endl;
91          }
92      }
93      return 0;
94  }
```

保证输入的 n,m 均不超过 10^5，且有 $1\le l\le r\le n$，$|w|\le 10^9$。试完成以下判断题和单选题：

- 判断题

28. 该程序的功能如下文所述：（　　）
读入一个数组[a_1,a_2,...,a_n]，并执行 m 次操作：
1 l r w：将 $a_l,a_{l+1},...,a_r$ 加上 w；-2 l r w：计算 $a_l+a_{l+1}+...+a_r$ 的值并输出。

29. 如果将主函数里的程序片段 root=CirnoTree::build1(1,n);修改成 root=CirnoTree::build2(1,3*n);，则程序针对任意合法输入数据的输出结果不会发生任何改变。（　　）

30. 如果在 build 操作以后，执行函数片段 CirnoTree::query(root,-142857)，程序会因为发生数组越界而非正常退出。（　　）

- 单选题

31. （2 分）当读入的 n 充分大时，CirnoTree 的大小（即在主程序执行完 build 操作后变量 siz 的值）大约为（　　）。

A. 2n　　B. $n\log_2 n$
C. 1.75n　　D. 1.5n

32. 对于该数据结构，调用一次 build 函数的**最好**时间复杂度为（　）；调用一次 modify 或 query 函数的**最坏**时间复杂度为（　　）。

A. $\Theta(n);\Theta(\log n)$　　B. $\Theta(n\log n);\Theta(\log^2 n)$
C. $\Theta(n);\Theta(\log^2 n)$　　D. $\Theta(n\log n);\Theta(n)$

33. 对于如下输入数据，在程序主要片段运行结束、执行到 return 0 之前，below_tag[8]和 right_tag[8]的值分别是（）：

```
8 3
0 0 0 0 0 0 0 0
1 1 8 3
1 2 6 2
2 5 8
```

A. 2，0　　B. 0，0
C. 5，0　　D. 3，2

三、完善程序（单选题，每小题 3 分，共计 30 分）

(1)（异或和）给定序列 a_n，求其所有子区间异或和的和。某个区间的异或和的意思是将这个区间内的所有数字分别进行异或计算。其中 $n \le 10^5$，$0 \le a_i \le 10^9$。

提示：对每一位独立计算，对右端点扫描线，并用异或前缀和辅助统计。

试补全程序。

```
01  #include<bits/stdc++.h>
02  using namespace std;
03  const int N = 1e5 + 7;
04  int n, a[N], cnt[2];
05  long long ans;
06  int main () {
07      cin >> n;
08      for (int i = 1; i <= n; i ++)
09          cin >> a[i], ①;
10      for (int bit = ②; ③; bit --) {
11          cnt[0] = cnt[1] = 0;
12          for (int i = 0; i <= n; i ++) {
13              cnt[④] ++;
14              ans += 1LL* ⑤;
15          }
16      } cout << ans;
17  }
```

34. ① 处应填（　　）。

A. a[i - 1] ^= a[i]　　B. a[i] ^= a[i - 1]
C. a[i - 1] += a[i]　　D. a[i] += a[i - 1]

35. ② 处应填（　　）。

A. n - 1　　B. 29
C. n　　D. n + 1

36. ③ 处应填（　　）。

A. bit　　B. bit >= n
C. bit - 1　　D. ~bit

37. ④ 处应填（　　）。

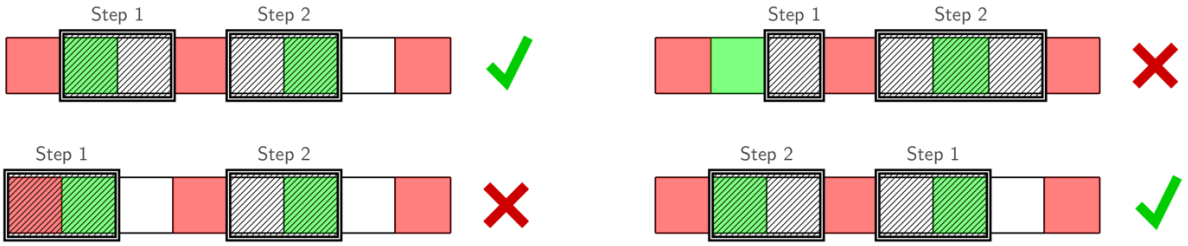
A. (a[i] >> bit) & 1　　B. a[i] & (1 << bit)
C. a[i] & (1LL << bit)　　D. (a[i] >> bit) ^ 1

38. ⑤ 处应填（　　）。

A. `cnt[(a[i] >> bit) & 1] << i`
B. `cnt[(a[i] >> bit) & 1 ^ 1] << bit`
C. `cnt[(a[i] >> bit) & 1] << bit`
D. `cnt[(a[i] >> bit) & 1 ^ 1] << i`

(2)（覆盖）给定序列$\{a_n\}$（$1\le n\le 100$，$a_i\in\{0,1,2\}$），描述 n 个格子的状态。若a_i为 0，则表示覆不覆盖均可；若为 1，则表示一定不能覆盖；若为 2，则表示一定要被覆盖。询问依次进行 m 次操作（$1\le m\le 10^9$），每次选定一个区间[l,r]的格子将其覆盖，使得满足a_n数组的所有限制条件，有多少种方案？两种方案不同，当且仅当**存在**某次操作覆盖的区间不同。如果格子可以被覆盖，则可以被覆盖多次。

如图例（第一张图和第四张图算不同方案，因为步骤 1 覆盖的区间不同。）：



提示：考虑容斥原理，使用动态规划求出每种权值的贡献次数。

试补全程序。

```
01  #include<iostream>
02  using namespace std;
03  const int MAXN= 100 + 3;
04  const int MAXM= 1e4 + 3;
05  const int MOD = 1e9 + 7;
06  int F[2][MAXN][MAXM], A[MAXN];
07  int power(int a, int b){
08      int r = 1;
09      while(b){
10          if(b & 1) r = 1ll * r * a % MOD;
11          b >>= 1,  ①;
12      }
13      return r;
14  }
15  int main(){
16      int n, m;
17      cin >> n >> m;
18      int h = n * (n + 1) / 2;
19      for(int i = 1;i <= n;i ++)
20          cin >> A[i];
```

```
21      ②;
22      F[0][0][0] = 1;
23      for(int i = 1;i <= n + 1;i ++)
24        if(A[i] == 1 || A[i] == 2){
25          for(int j = i - 1;j >= 0;j --){
26              int c = ③;
27              int g = ④;
28              for(int k = h;k >= g;-- k){
29                  F[0][i][k]=(F[0][i][k]+F[0^c][j][k-g])%MOD;
30                  F[1][i][k]=(F[1][i][k]+F[1^c][j][k-g])%MOD;
31              }
32              if(A[j] == 1) break;
33          }
34        }
35      int ans = 0;
36      for(int i = 0;i <= h;i ++){
37          ans = (ans + 1ll * F[0][n+1][i] * power(i, m)) % MOD;
38          ans = (ans + 1ll * ⑤ * power(i, m)) % MOD;
39      }
40      cout << ans << endl;
41      return 0;
42  }
```

39. ① 处应填（　　）。

A. r = 1ll * r * r % MOD
B. a = 1ll * a * a % MOD
C. r = 1ll * r * a % MOD
D. a = 1ll * r * a % MOD

40. ② 处应填（　　）。

A. A[0] = 1, A[n + 1] = 1
B. A[0] = 1, A[n + 1] = 2
C. A[0] = 2, A[n + 1] = 1
D. A[0] = 2, A[n + 1] = 2

41. ③ 处应填（　　）。

A. (A[i] == 2) ^ (A[j] == 2)
B. A[j] == 2
C. A[i] == 2
D. (A[i] == 2) ^ (A[j] == 2)

42. ④ 处应填（　　）。

A. !(i - j) * (i - j + 1)
B. (i - j) * (i - j + 1) / 2
C. (i - j - 1) * (i - j) / 2
D. (i - j) * (i - j)

43. ⑤ 处应填（　　）。

A. (F[1][n + 1][i]-1+MOD)　　B. F[0][n + 1][i]
C. (MOD - F[1][n + 1][i])　　D. (MOD - F[0][n + 1][i])