2024 CCF 非专业级别软件能力认证第一轮
（CSP-S1）提高级 C++语言试题

认证时间：2024 年 9 月 21 日 14:30~16:30

考生注意事项：

- 试题纸共有 16 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. 在 Linux 系统中，如果你想显示当前工作目录的路径，应该使用哪个命令？（ ）

A. pwd

B. cd

C. ls

D. echo

2. 假设一个长度为 n 的整数数组中每个元素值互不相同，且这个数组是无序的。要找到这个数组中最大元素的时间复杂度是多少？（ ）

A. O(n)

B. O(log n)

C. O(n log n)

D. O(1)

3. 在 C++中，以下哪个函数调用会造成栈溢出？（ ）

A. `int foo() { return 0; }`

B. `int bar() { int x = 1; return x; }`

C. `void baz() { int a[1000]; baz(); }`

D. `void qux() { return; }`

4. 在一场比赛中，有 10 名选手参加，前三名将获得金、银、铜牌。若不允许并列，且每名选手只能获得一枚奖牌，则不同的颁奖方式共有多少种？（ ）

A. 120

B. 720

C. 504

D. 1000

5. 下面哪个数据结构最适合实现先进先出（FIFO）的功能？（ ）

A. 栈

B. 队列

C. 线性表

D. 二叉搜索树

6. 已知$f(1) = 1$，且对于$n \geq 2$有$f(n) = f(n-1) + f(\lfloor n/2 \rfloor)$，则$f(4)$的值为：（ ）

A. 4

B. 5

C. 6

D. 7

7. 假设有一个包含 n 个顶点的无向图，且该图是欧拉图。以下关于该图的描述中哪一项不一定正确？（ ）

A. 所有顶点的度数均为偶数

B. 该图连通

C. 该图存在一个欧拉回路

D. 该图的边数是奇数

8. 对数组进行二分查找的过程中，以下哪个条件必须满足？（ ）

A. 数组必须是有序的

B. 数组必须是无序的

C. 数组长度必须是 2 的幂

D. 数组中的元素必须是整数

9. 考虑一个自然数 n 以及一个模数 m，你需要计算 n 的逆元（即 n 在模 m 意义下的乘法逆元）。下列哪种算法最为适合？（ ）

A. 使用暴力法依次尝试

B. 使用扩展欧几里得算法

C. 使用快速幂法

D. 使用线性筛法

10. 在设计一个哈希表时，为了减少冲突，需要使用适当的哈希函数和冲突解决策略。已知某哈希表中有 n 个键值对，表的装载因子为 α（$0 < \alpha \leq 1$）。在使用开放地址法解决冲突的过程中，最坏情况下查找一个元素的时间复杂度为（ ）

A. O(1)

B. O(log n)

C. O(1 / (1 - α))

D. O(n)

11. 假设有一棵 h 层的完全二叉树，该树最多包含多少个结点？

A. 2^h-1

B. 2^(h+1)-1

C. 2^h

D. 2^(h+1)

12. 设有一个 10 个顶点的完全图，每两个顶点之间都有一条边。有多少个长度为 4 的环？（ ）

A. 120

B. 210

C. 630

D. 5040

13. 对于一个整数 n，定义 f(n)为 n 的各位数字之和。问使 f(f(x))=10 的最小自然数 x 是

多少？（ ）

A. 29

B. 199

C. 299

D. 399

14. 设有一个长度为 n 的 01 字符串，其中有 k 个 1，每次操作可以交换相邻两个字符。在最坏情况下将这 k 个 1 移到字符串最右边所需要的交换次数是多少？（ ）

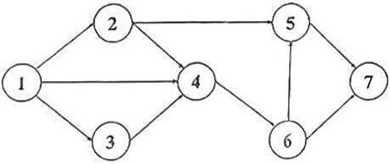
A. k

B. k*(k-1)/2

C. (n-k)*k

D. (2n-k-1)*k/2

15. 如图是一张包含 7 个顶点的有向图，如果要删除其中一些边，使得从节点 1 到节点 7 没有可行路径，且删除的边数最少，请问总共有多少种可行的删除边的集合？（ ）



A. 1

B. 2

C. 3

D. 4

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）

```
#include <iostream>
using namespace std;

const int N = 1000;
int c[N];

int logic(int x, int y) {
    return (x & y) ^ ((x ^ y) | (~x & y));
}
void generate(int a, int b, int *c) {
    for (int i = 0; i < b; i++)
        c[i] = logic(a, i) % (b + 1);
    }
}
void recursion(int depth, int *arr, int size) {
    if (depth <= 0 || size <= 1) return;
    int pivot = arr[0];
    int i = 0, j = size - 1;
    while (i <= j) {
        while (arr[i] < pivot) i++;
        while (arr[j] > pivot) j--;
        if (i <= j) {
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
            i++; j--;
        }
    }
    recursion(depth - 1, arr, j + 1);
    recursion(depth - 1, arr + i, size - i);
}

int main() {
    int a, b, d;
    cin >> a >> b >> d;
    generate(a, b, c);
    recursion(d, c, b);
    for (int i = 0; i < b; ++i) cout << c[i] << " ";
```

```
    cout << endl;
}
```

◉ 判断题

16．当 1000≥d≥b 时，输出的序列是有序的。（　）

17．当输入"5 5 1"时，输出为"1 1 5 5 5"。（　）

18．假设数组 c 长度无限制，该程序所实现的算法的时间复杂度是 O(b)的。（　）

◉ 单选题

19．函数 int logic(int x, int y)的功能是（　）

A．按位与

B．按位或

C．按位异或

D．以上都不是

20．（4 分）当输入为“10 100 100”时，输出的第 100 个数是（　）

A．91

B．94

C．95

D．98

(2)

```
#include <iostream>
#include <string>
using namespace std;

const int P = 998244353, N = 1e4+10, M = 20;
int n, m;
string s;
int dp[1<<M];

int solve() {
    dp[0] = 1;
    for (int i = 0; i < n; ++i) {
        for (int j = (1<<(m-1))-1; j >= 0; --j) {
```

```cpp
            int k = (j<<1)|(s[i]-'0');
            if (j != 0 || s[i] == '1')
                dp[k] = (dp[k] + dp[j]) % P;


    int ans = 0;
    for (int i = 0; i < (1<<m); ++i) {
        ans = (ans + 1ll * i * dp[i]) % P;
    }
    return ans;
}
int solve2() {
    int ans = 0;
    for (int i = 0; i < (1<<n); ++i) {
        int cnt = 0;
        int num = 0;
        for (int j = 0; j < n; ++j) {
            if (i & (1<<j)) {
                num = num * 2 + (s[j]-'0');
                cnt++;

        }
        if (cnt <= m) (ans += num) %= P;
    }
    return ans;


int main() {
    cin >> n >> m;
    cin >> s;
    if (n <= 20) {
        cout << solve2() << endl;
    }
    cout << solve() << endl;
    return 0;
}
```

假设输入的 s 是包含 n 个字符的 01 串，完成下面的判断题和单选题：

- 判断题

21. 假设数组 dp 长度无限制，函数 solve()所实现的算法的时间复杂度是 O(n*2^m)。（ ）

22. 输入“11 2 10000000001”时，程序输出两个数 32 和 23。（ ）

23. （2 分）在 n≤10 时，solve()的返回值始终小于 4^{10}。（ ）

- 单选题

24. 当 n = 10 且 m = 10 时，有多少种输入使得两行的结果完全一致？（ ）

A. 1024

B. 11

C. 10

D. 0

25. 当 n <= 6 时，solve()的最大可能返回值为（ ）

A. 65

B. 211

C. 665

D. 2059

26. 若 n = 8，m = 8，solve 和 solve2 的返回值的最大可能的差值为（ ）

A. 1477

B. 1995

C. 2059

D. 2187

（3）

```
#include <iostream>
#include <cstring>
#include <algorithm>
using namespace std;

const int maxn = 1000000+5;
const int P1 = 998244353, P2 = 1000000007;
const int B1 = 2, B2 = 31;
const int K1 = 0, K2 = 13;

typedef long long ll;

```

```
int n;
bool p[maxn];
int p1[maxn], p2[maxn];

struct H {
    int h1, h2, l;
    H(bool b = false) {
        h1 = b + K1;
        h2 = b + K2;
        l = 1;
    }
    H operator + (const H & h) const {
        H hh;
        hh.l = l + h.l;
        hh.h1 = (1ll * h1 * p1[h.l] + h.h1) % P1;
        hh.h2 = (1ll * h2 * p2[h.l] + h.h2) % P2;
        return hh;
    }
    bool operator == (const H & h) const {
        return l == h.l && h1 == h.h1 && h2 == h.h2;
    }
    bool operator < (const H & h) const {
        if (l != h.l) return l < h.l;
        else if (h1 != h.h1) return h1 < h.h1;
        else return h2 < h.h2;
    }
} h[maxn];

void init() {
    memset(p, 1, sizeof(p));
    p[0] = p[1] = false;
    p1[0] = p2[0] = 1;
    for (int i = 1; i <= n; ++i) {
        p1[i] = (1ll * B1 * p1[i-1]) % P1;
        p2[i] = (1ll * B2 * p2[i-1]) % P2;
        if (!p[i]) continue;
        for (int j = 2 * i; j <= n; j += i) {
            p[j] = false;
        }
    }
}

```

```
int solve() {
    for (int i = n; i; --i) {
        h[i] = H(p[i]);
        if (2 * i + 1 <= n) {
            h[i] = h[2 * i] + h[i] + h[2 * i + 1];
        } else if (2 * i <= n) {
            h[i] = h[2 * i] + h[i];
        }
    }

    cout << h[1].h1 << endl;
    sort(h + 1, h + n + 1);
    int m = unique(h + 1, h + n + 1) - (h + 1);
    return m;
}

int main() {
    cin >> n;
    init();
    cout << solve() << endl;
}
```

- 判断题

27. 假设程序运行前能自动将 maxn 改为 n+1，所实现的算法的时间复杂度是 O(n log n)。（ ）

28. 时间开销的瓶颈是 init()函数。（ ）

29. 若修改常数 B1 或 K1 的值，该程序可能会输出不同的结果。（ ）

- 单选题

30. 在 solve()函数中，h[]的合并顺序可以看作是：（ ）

A. 二叉树的 BFS 序

B. 二叉树的先序遍历

C. 二叉树的中序遍历

D. 二叉树的后序遍历

31. 输入“10”，输出的第一行是？（ ）

A. 83

B. 424

C. 54

D. 110101000

32.（4 分）输入“16”，输出的第二行是？（C）

A. 7

B. 9

C. 10

D. 12

三、完善程序（单选题，每小题 3 分，共计 30 分）

(1)（序列合并）有两个长度为 N 的单调不降序列 A 和 B，序列的每个元素都是小于 10^9 的非负整数。在 A 和 B 中各取一个数相加可以得到 N^2 个和，求其中第 K 小的和。上述参数满足 N <= 10^5 和 1 <= K <= N^2。

```
#include <iostream>
using namespace std;

const int maxn = 100005;

int n;
long long k;
int a[maxn], b[maxn];

int* upper_bound(int *a, int *an, int ai) {
    int l = 0, r = ____①____;
    while (l < r) {
        int mid = (l+r)>>1;
        if (____②____) {
            r = mid;
        } else {
            l = mid + 1;
        }
    }
    return ____③____;
}

long long get_rank(int sum) {
    long long rank = 0;
    for (int i = 0; i < n; ++i) {
        rank += upper_bound(b, b+n, sum - a[i]) - b;
    }
    return rank;
}

int solve() {
    int l = 0, r = ____④____;
    while (l < r) {
        int mid = ((long long)l+r)>>1;
```

```
        if (___⑤___) {
            l = mid + 1;
```

```
        } else {
            r = mid;
        }
    }
    return l;
}

int main() {
    cin >> n >> k;
    for (int i = 0; i < n; ++i) cin >> a[i];
    for (int i = 0; i < n; ++i) cin >> b[i];
    cout << solve() << endl;
}
```

33.①处应填

A. an-a
B. an-a-1
C. ai
D. ai+1

34.②处应填

A. a[mid] > ai
B. a[mid] >= ai
C. a[mid] < ai
D. a[mid] <= ai

35.③处应填

A. a+l
B. a+l+1
C. a+l-1
D. an-l

36.④处应填

A. a[n-1]+b[n-1]
B. a[n]+b[n]
C. 2 * maxn
D. maxn

37.⑤处应填

A. get_rank(mid) < k　　　　B. get_rank(mid) <= k

C. get_rank(mid) > k　　　　D. get_rank(mid) >= k

---

(2)（次短路）已知一个有 n 个点 m 条边的有向图 G，并且给定图中的两个点 s 和 t，求次短路（长度严格大于最短路的最短路径）。如果不存在，输出一行“-1”。如果存在，输出两行，第一行表示次短路的长度，第二行表示次短路的一个方案。

```
#include <cstdio>
#include <queue>
#include <utility>
#include <cstring>
using namespace std;

const int maxn = 2e5+10, maxm = 1e6+10, inf = 522133279;

int n, m, s, t;
int head[maxn], nxt[maxm], to[maxm], w[maxm], tot = 1;
int dis[maxn<<1], *dis2;
int pre[maxn<<1], *pre2;
bool vis[maxn<<1];

void add(int a, int b, int c) {
    ++tot;
    nxt[tot] = head[a];
    to[tot] = b;
    w[tot] = c;
    head[a] = tot;
}

bool upd(int a, int b, int d, priority_queue<pair<int, int>> &q) {
    if (d >= dis[b]) return false;
    if (b < n) ____①____;
    q.push(____②____);
    dis[b] = d;
    pre[b] = a;
    return true;
}

void solve() {
```

```
    priority_queue<pair<int, int> > q;
    q.push(make_pair(0, s));
    memset(dis, ____③____, sizeof(dis));
    memset(pre, -1, sizeof(pre));
    dis2 = dis+n;
    pre2 = pre+n;
```

```
    dis[s] = 0;
    while (!q.empty()) {
        int aa = q.top().second; q.pop();
        if (vis[aa]) continue;
        vis[aa] = true;
        int a = aa % n;
        for (int e = head[a]; e; e = nxt[e]) {
            int b = to[e], c = w[e];
            if (aa < n) {
                if (!upd(a, b, dis[a]+c, q))
                    ____④____;
            } else {
                upd(n+a, n+b, dis2[a]+c, q);
            }




void out(int a) {
    if (a != s) {
    if (a < n) out(pre[a]);
    else out(____⑤____);
    }
    printf("%d%c", a%n+1, " \n"[a == n+t]);
}

int main() {
    scanf("%d%d%d%d", &n, &m, &s, &t);
    s--, t--;
    for (int i = 0; i < m; ++i) {
```

```
        int a, b, c;
        scanf("%d%d%d", &a, &b, &c);
        add(a-1, b-1, c);
    }
    solve();
    if (dis2[t] == inf) puts("-1");
    else {
        printf("%d\n", dis2[t]);
        out(n+t);
    }

```

38.①处应填（　）

A. upd(pre[b], n+b, dis[b], q)　B. upd(a, n+b, d, q)

C. upd(pre[b], b, dis[b], q)　D. upd(a, b, d, q)

39.②处应填（　）

A. make_pair(-d, b)　B. make_pair(d, b)

C. make_pair(b, d)　D. make_pair(-b, d)

40.③处应填（　）

A. 0xff　B. 0x1f

C. 0x3f　D. 0x7f

41.④处应填（　）

A. upd(a, n+b, dis[a]+c, q)　B. upd(n+a, n+b, dis2[a]+c, q)

C. upd(n+a, b, dis2[a]+c, q)　D. upd(a, b, dis[a]+c, q)

42.⑤处应填（　）

A. pre2[a%n]　B. pre[a%n]

C. pre2[a]　D. pre[a%n]+1