CCF 编程能力等级认证
Grade Examination of Software Programming

C++ 八级

2026 年 06 月

1 单选题（每题 2 分，共 30 分）

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

第 1 题 从 7 本不同的算法书和 5 本不同的数学书中选出 4 本，要求两类书都至少选 1 本，共有（ ）种不同选法。

- **A.** 420
- **B.** 455
- **C.** 465
- **D.** 495

第 2 题 6 个人排成一排照相，其中甲、乙两人不能相邻，共有（ ）种不同排法。

- **A.** 240
- **B.** 480
- **C.** 600
- **D.** 720

第 3 题 展开式 $(x^2-\frac{1}{x})^6$ 中，常数项的系数为（ ）。

- **A.** 6
- **B.** 12
- **C.** 15
- **D.** 20

第 4 题 下面代码用于预处理组合数，横线处应填入的是（ ）。

```
for (int i = 0; i <= n; i++) {
    c[i][0] = c[i][i] = 1;
    for (int j = 1; j < i; j++)
        c[i][j] = __________;
}
```

- **A.** `c[i - 1][j - 1] + c[i - 1][j]`
- **B.** `c[i][j - 1] + c[i - 1][j - 1]`
- **C.** `c[i - 1][j] + c[i][j + 1]`
- **D.** `c[i][j - 1] * c[i - 1][j]`

第 5 题 下列程序输出的值为（ ）。

```cpp
#include <iostream>
using namespace std;
long long qpow(long long a, long long b, long long mod) {
    long long ans = 1 % mod;
    while (b) {
        if (b & 1)
            ans = ans * a % mod;
        a = a * a % mod;
        b >>= 1;
    }
    return ans;
}
int main() {
    cout << qpow(3, 20, 17) << endl;
    return 0;
}
```

- [ ] **A.** 1
- [ ] **B.** 4
- [ ] **C.** 13
- [ ] **D.** 16

第 6 题 归并排序每次把长度为 n 的序列分成两个规模约为 $n/2$ 的子序列，递归排序后再用线性时间合并。该算法的时间复杂度通常为（ ）。

- [ ] **A.** $O(n)$
- [ ] **B.** $O(n^2)$
- [ ] **C.** $O(\log n)$
- [ ] **D.** $O(n \log n)$

第 7 题 在平面直角坐标系中，三角形三个顶点为 $A(1,1)$、$B(5,2)$、$C(3,6)$，该三角形面积为（ ）。

- [ ] **A.** 9
- [ ] **B.** 10
- [ ] **C.** 12
- [ ] **D.** 18

第 8 题 某程序需要判断点 $P(x,y)$ 是否在以原点为圆心、半径为 5 的圆内或圆上。下列判断条件正确的是（ ）。

- [ ] **A.** `x * x + y * y <= 25`
- [ ] **B.** `abs(x) + abs(y) <= 5`
- [ ] **C.** `x * x - y * y <= 25`
- [ ] **D.** `x + y <= 5`

第 9 题 某无向带权图有边 $(1,2,4)$、$(1,3,2)$、$(2,3,1)$、$(2,4,5)$、$(3,4,8)$、$(3,5,10)$、$(4,5,2)$。该图最小生成树的总权值为（ ）。

- [ ] **A.** 7
- [ ] **B.** 8
- [ ] **C.** 9
- [ ] **D.** 10

第 10 题 有向非负权图边为 $1 \to 2(3)$、$2 \to 4(4)$、$1 \to 3(10)$、$3 \to 4(1)$、$2 \to 3(2)$。使用 Dijkstra 算法从 `1` 号顶点出发到 `4` 号顶点的最短距离为（ ）。

- [ ] **A.** 6
- [ ] **B.** 7
- [ ] **C.** 8
- [ ] **D.** 11

第 11 题 下列代码片段的时间复杂度为（ ）。

```
long long s = 0;
for (int i = 1; i <= n; i++) {
    for (int j = 1; j * j <= n; j++) {
        s += i + j;
    }
}
```

- [ ] **A.** $O(n)$
- [ ] **B.** $O(n \log n)$
- [ ] **C.** $O(n\sqrt{n})$
- [ ] **D.** $O(n^2)$

第 12 题 某优化问题的答案是 $[1, M]$ 内的整数，存在单调判定函数 `check(x)`，且每次判定的时间复杂度为 $O(n)$。使用二分答案求最小可行值，整体时间复杂度通常为（ ）。

- [ ] **A.** $O(nM)$
- [ ] **B.** $O(n \log M)$
- [ ] **C.** $O(M \log n)$
- [ ] **D.** $O(n + M)$

第 13 题 下列线性筛的代码片段中，当枚举到质数 `p` 且 `i % p == 0` 时，使用 `break;` 语句停止继续枚举。这样做的主要目的是（ ）。

```
for (int i = 2; i <= n; ++i) {
    if (!is_composite[i])
        primes.push_back(i);
    for (int p : primes) {
        if (i * p > n)
            break;
        is_composite[i * p] = true;
        if (i % p == 0)
            break;          // 这条语句的目的是?
    }
}
```

- [ ] **A.** 保证递归深度不超过 $O(\log n)$。
- [ ] **B.** 保证每个合数只被它的最小质因子筛去一次。
- [ ] **C.** 保证每个素数都被标记为合数。
- [ ] **D.** 把筛法时间复杂度提高到 $O(n \log n)$。

第 14 题 在 C++ 中，关于类的继承和构造、析构顺序，下列说法正确的是（ ）。

- [ ] **A.** 派生类可以直接访问基类的 `private` 成员。
- [ ] **B.** 基类的 `protected` 成员在私有继承后会变成派生类的 `public` 成员。
- [ ] **C.** 创建派生类对象时，会先调用基类构造函数，再调用派生类构造函数。
- [ ] **D.** 销毁派生类对象时，会先调用基类析构函数，再调用派生类析构函数。

第 15 题 将 4 个元素按 $1, 2, 3, 4$ 的顺序入栈，在该过程中可随时插入出栈操作。下列序列中不可能作为出栈序列的是（ ）。

- [ ] **A.** $1, 2, 3, 4$
- [ ] **B.** $2, 1, 4, 3$
- [ ] **C.** $3, 2, 1, 4$
- [ ] **D.** $3, 1, 2, 4$

2 判断题（每题 2 分，共 20 分）

题号	1	2	3	4	5	6	7	8	9	10
答案	√	√	×	×	√	×	√	√	×	√

第 1 题 若一项任务可从两种互斥的方案中选择一种完成，其中，方案 A 有 m 种做法，方案 B 有 n 种做法，则总做法数为 $m + n$。

第 2 题 将 n 个不同元素围成一圈，若只把旋转视为同一种排法、翻转仍视为不同排法，则方案数为 $(n-1)!$。

第 3 题 从 n 个不同元素中可重复地选取 k 个且不考虑顺序，方案数为 $C(n+k, k)$。

第 4 题 杨辉三角中的组合数满足 $C(n, k) = C(n-1, k) + C(n-2, k)$。

第 5 题 快速幂通过二进制拆分指数，可以在 $O(\log b)$ 时间内计算 $a^b \bmod m$。

第 6 题 只要图中不存在负权环，Dijkstra 算法就一定能正确处理带负权边的图。

第 7 题 若一张连通无向图所有边权两两不同，则它的最小生成树一定唯一。

第 8 题 判断点 (x, y) 是否在以原点为圆心、半径为 r 的圆内或圆上时，可以比较 $x^2 + y^2$ 与 r^2，不必先开平方。

第 9 题 若能写出判定函数 `check(x)`，表示“答案为 `x` 时是否可行”，即使 `check(x)` 不满足单调性，也一定可以使用二分答案求最优解。

第 10 题 归并排序是一种稳定排序算法，常见实现的时间复杂度为 $O(n \log n)$。

3 编程题（每题 25 分，共 50 分）

3.1 编程题 1

- **试题名称**：线网建设
- **时间限制**：1.0 s
- **内存限制**：512.0 MB

3.1.1 题目描述

A 市有 n 座基站需要通过线网互相连接。第 i 座基站位于二维平面上坐标 (x_i, y_i) 处。

第 i 座基站与第 j 座基站之间的距离定义为 $\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$。

如果两座基站之间的距离不超过给定的整数 l，那么可以修建连接这两座基站的线路，线路长度为基站间的距离。

如果从一座基站出发，经过一系列线网中的线路可以到达另一座基站，则称这两座基站是互相连接的。

请问使得 n 座基站两两之间都互相连接，需要修建的线路总长度最小是多少？如果不能修建满足条件的线网，则输出 `Impossible`。

3.1.2 输入格式

第一行，两个正整数 n, l，分别表示基站数量与线路长度上限。

接下来 n 行，每行两个整数 x_i, y_i，表示基站的坐标。

3.1.3 输出格式

输出一行。如果能修建满足条件的线网，则输出需要修建的最小线路总长度，保留两位小数。否则输出 `Impossible`。

3.1.4 样例

3.1.5 输入样例 1

```
4 2
1 0
-1 -1
0 0
1 1
```

3.1.6 输出样例 1

```
3.41
```

3.1.7 输入样例 2

```
4 1
1 0
-1 -1
0 0
1 1
```

3.1.8 输出样例 2

```
Impossible
```

3.1.9 数据范围

对于 40% 的测试点，保证 $1 \leq n \leq 100$。

对于所有测试点，保证 $1 \leq n \leq 500$，$1 \leq l \leq 100$，$-100 \leq x_i, y_i \leq 100$。

3.1.10 参考程序

```cpp
#include <iostream>
#include <cstdio>
#include <algorithm>
#include <cmath>
using namespace std;

const int N = 510;
const int E = N * N;

int n, l;
int x[N], y[N];
int p[E], u[E], v[E], cnt;
int f[N], t = 0;
double d[E], ans = 0;

int getf(int u) {
    return f[u] ? f[u] = getf(f[u]) : u;
}

bool cmp(int a, int b) {
    return d[a] < d[b];
}

int main() {
    cin >> n >> l;
    for (int i = 1; i <= n; i++)
        cin >> x[i] >> y[i];
    for (int i = 1; i <= n; i++)
        for (int j = i + 1; j <= n; j++) {
            int dx = x[i] - x[j], dy = y[i] - y[j];
            if (dx * dx + dy * dy > l * l)
                continue;
            cnt++;
            p[cnt] = cnt;
            u[cnt] = i;
            v[cnt] = j;
            d[cnt] = sqrt(dx * dx + dy * dy);
        }
    sort(p + 1, p + cnt + 1, cmp);
    for (int i = 1; i <= cnt; i++) {
        int pu = u[p[i]], pv = v[p[i]];
        if (getf(pu) == getf(pv))
            continue;
        t++;
        ans += d[p[i]];
        f[getf(pu)] = pv;
    }
    if (t == n - 1)
        printf("%.2lf\n", ans);
    else
        printf("Impossible\n");
    return 0;
}
```

3.2 编程题 2

- **试题名称**：堆石子
- **时间限制**：1.0 s
- **内存限制**：512.0 MB

3.2.1 题目描述

有 m 堆石子，编号为 $1,2,\cdots,m$，其石子数量分别记为 $a_1,a_2,\cdots,a_m$。

现在要求第 1 堆石子恰有 n 个（即 $a_1=n$），并且此后每堆石子的数量严格小于前一堆，即 $a_i<a_{i-1}$（$2\le i\le m$）。此外，每堆至少需要有一个石子，即 $a_i\ge 1$（$1\le i\le m$）。

在总石子数量不设限制的情况下，给定 $m\ge 2,n\ge 1$，有多少个满足要求的石子堆放方案?

两个方案不同，当且仅当，两个方案中至少有一堆石子数量不同。

如果不存在满足要求的方案，输出 0。由于方案数可能很大，请输出方案数对 10^9+7 取模后的结果。

3.2.2 输入格式

输入一行两个正整数 m 和 n。

3.2.3 输出格式

输出一个整数，表示总方案数对 10^9+7 取模后的结果。

3.2.4 样例

3.2.5 输入样例 1

```
3 5
```

3.2.6 输出样例 1

```
6
```

3.2.7 样例解释 1

有 $(5,4,3)$，$(5,4,2)$，$(5,4,1)$，$(5,3,2)$，$(5,3,1)$ 和 $(5,2,1)$ 共计 6 种方案。

3.2.8 数据范围

数据点编号	数据范围	特殊性质
1,2	$2\le m\le 100,1\le n\le 100$	$0\le n-m\le 5$
3,4,5	$2\le m\le 100,1\le n\le 10^8$	无
6,7,8,9,10	$2\le m\le 10^5,1\le n\le 10^8$	无

3.2.9 参考程序

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

const int MOD = (int)1e9 + 7;

int qpow(int base, int exp) {
    if (!exp) return 1;
    if (exp & 1) return (long long)base * qpow((long long)base * base % MOD, exp >> 1) % MOD;
    return qpow((long long)base * base % MOD, exp >> 1);
}

int comb(int n, int m) {
    if (m > n) return 0;
    int ans = 1;
    for (int i = 0; i < m; ++i) {
        ans = (long long)ans * (n - i) % MOD;
        ans = (long long)ans * qpow(i + 1, MOD - 2) % MOD;
    }
    return ans;
}

int main() {
    int m, n;
    cin >> m >> n;
    cout << comb(n - 1, m - 1) << endl;
    return 0;
}
```