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

C++ 八级

2023 年 12 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	C	A	D	B	C	D	A	C	A	C	D	B	B	A	C

第 1 题 小杨要从A城到B城，又想顺路游览一番。他有两个选项：1、坐高铁路到C城游览，再坐高铁或飞机到B城；2、坐船到D城游览，再坐船、高铁或飞机到B城。请问小杨从A城到B城共有几种交通方案可以选择？（ ）。

- ☐ **A.** 2
- ☐ **B.** 3
- ☐ **C.** 5
- ☐ **D.** 6

第 2 题 以下哪个函数声明是符合语法的，且在调用时可以将二维数组的名字作为实际参数传递给形式参数 `a`？（ ）。

- ☐ **A.** `void QuickSort(int a[][10], int n);`
- ☐ **B.** `void QuickSort(int a[5][], int m);`
- ☐ **C.** `void QuickSort(int a[][], int n, int m);`
- ☐ **D.** `void QuickSort(int ** a, int n, int m);`

第 3 题 下面有关C++类和对象的说法，错误的是（ ）。

- ☐ **A.** 对象的生命周期开始时，会执行构造函数。
- ☐ **B.** 对象的生命周期结束时，会执行析构函数。
- ☐ **C.** 类的析构函数可以为虚函数。
- ☐ **D.** 类的构造函数可以为虚函数。

第 4 题 使用邻接矩阵表达 `n` 个顶点的有向图，则该矩阵的大小为（ ）。

- ☐ **A.** $n \times (n+1)$
- ☐ **B.** $n \times n$
- ☐ **C.** $n \times (n-1)$
- ☐ **D.** $n \times (n-1)/2$

第 5 题 `5` 位同学排队，其中一位同学不能排在第一，则共有多少种可能的排队方式？（ ）。

- ☐ **A.** `5`
- ☐ **B.** `24`
- ☐ **C.** `96`
- ☐ **D.** `120`

第 6 题 一个无向图包含 `n` 个顶点，则其最小生成树包含多少条边？（ ）。

- ☐ **A.** $n-1$
- ☐ **B.** n
- ☐ **C.** $n+1$
- ☐ **D.** 最小生成树可能不存在。

第 7 题 已知三个 `double` 类型的变量 `a` 、 `b` 和 `theta` 分别表示一个三角形的两条边长及二者的夹角（弧度），则下列哪个表达式可以计算这个三角形的面积？（ ）。

- ☐ **A.** `a * b * sin(theta) / 2`
- ☐ **B.** `(a + b) * sin(theta) / 2`
- ☐ **C.** `a * b * cos(theta) / 2`
- ☐ **D.** `sqrt(a * a + b * b - 2 * a * b * cos(theta))`

第 8 题 对有 `n` 个元素的二叉排序树进行中序遍历，其时间复杂度是（ ）。

- ☐ **A.** $O(1)$
- ☐ **B.** $O(\log(n))$
- ☐ **C.** $O(n)$
- ☐ **D.** $O(n^2)$

第 9 题 假设输入参数 `m` 和 `n` 满足$m \leq n$，则下面程序的最差情况的时间复杂度为（ ）。

```
int gcd(int m, int n) {
    while (m > 0) {
        int t = m;
        m = n % m;
        n = t;
    }
    return n;
}
```

- ☐ **A.** $O(\log(n))$
- ☐ **B.** $O(n)$
- ☐ **C.** $O(n \times m)$
- ☐ **D.** $O(m \times \log(n))$

第 10 题 下面程序的时间复杂度为（ ）。

```cpp
long long power_mod(long long a, long long n, long long mod) {
    if (n == 0)
        return 1;
    a = a % mod;
    if (n == 1)
        return a;
    long long pw = power_mod(a, n / 2, mod);
    long long pw2 = pw * pw % mod;
    if (n % 2 == 0)
        return pw2;
    return pw2 * a % mod;
}
```

- ☐ **A.** $O(n)$
- ☐ **B.** $O(a^n)$
- ☐ **C.** $O(\log(n))$
- ☐ **D.** $O(\log(n) \times a)$

第 11 题 下面程序的时间复杂度为（ ）。

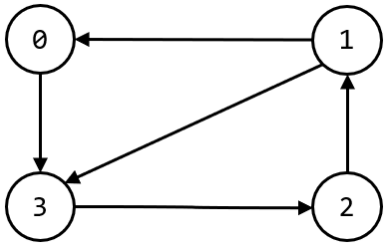
```cpp
int record_choose[MAXN][MAXM];
int choose(int n, int m) {
    if (m == 0 || m == n)
        return 1;
    if (record_choose[n][m] == 0)
        record_choose[n][m] = choose(n - 1, m - 1) + choose(n - 1, m);
    return record_choose[n][m];
}
```

- ☐ **A.** $O(2^n)$
- ☐ **B.** $O(2^m \times (n-m))$
- ☐ **C.** $O(C(n,m))$
- ☐ **D.** $O(m \times (n-m))$

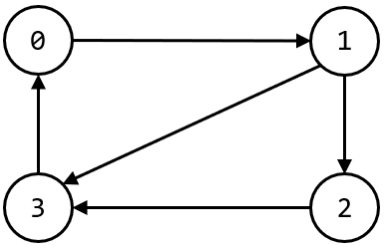
第 12 题 下面的程序使用出边的邻接表表达有向图，则下列选项中哪个是它表达的图？（ ）。

```cpp
#include <iostream>

struct Edge {
    int e;
    Edge * next;
};
struct Node {
    Edge * first;
};

int main() {
    Edge e[5] = {{1, nullptr}, {2, &e[2]},
                 {3, nullptr}, {3, nullptr}, {0, nullptr}};
    Node n[4] = {&e[0], &e[1], &e[3], &e[4]};
    ; // 其他处理
    return 0;
}
```

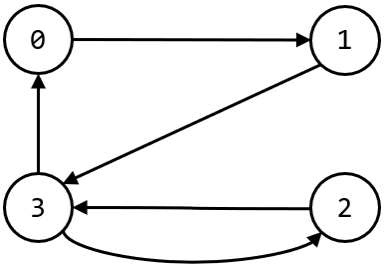
A.



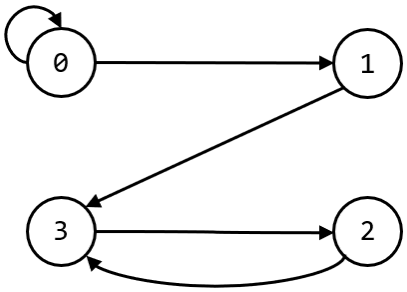
B.



C.



D.



第 13 题 下面程序的输出为（ ）。

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

int main() {
    int cnt = 0;
    for (int a = 1; a <= 10; a++)
        for (int b = 1; b <= 10; b++)
            for (int h = 1; h <= 10; h++)
                if ((a + b) * h == 20)
                    cnt++;
    cout << cnt << endl;
    return 0;
}
```

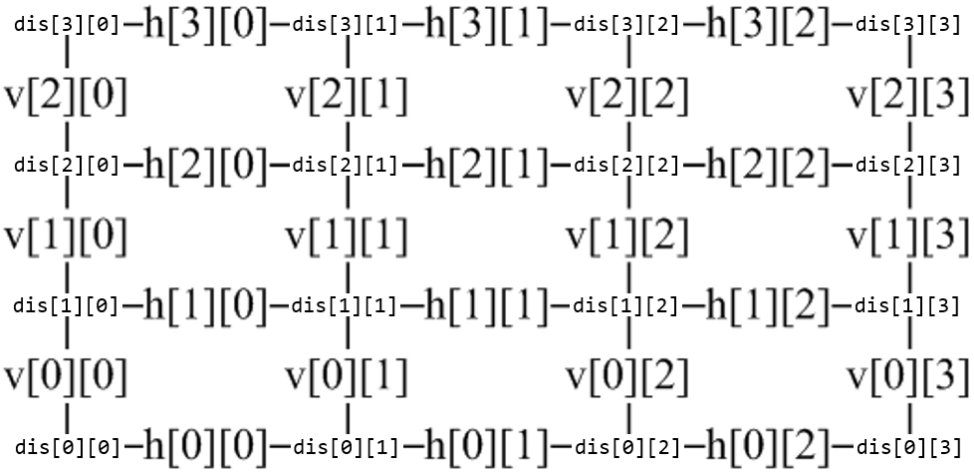
- A. 12
- B. 18
- C. 36
- D. 42

第 14 题 下面程序的输出为（ ）。

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

int main() {
    const int N = 30;
    int cnt = 0;
    for (int a = 1; a <= N; a++)
        for (int b = a; a + b <= N; b++)
            for (int c = b; a + b + c <= N; c++)
                if (a * a + b * b == c * c)
                    cnt++;
    cout << cnt << endl;
    return 0;
}
```

- A. 3
- B. 6
- C. 11
- D. 22

第 15 题 下面的程序中，二维数组 `h` 和 `v` 分别代表如下图所示的网格中的水平边的时间消耗和垂直边的时间消耗。程序使用动态规划计算从左下角到右上角的最小时间消耗，则横线处应该填写下列哪个选项的代码？（ ）。




```cpp
int dis[MAXY][MAXX];
int shortest_path(int x, int y) {
    dis[0][0] = 0;
    for (int i = 0; i < y; i++)
        dis[i + 1][0] = dis[i][0] + v[i][0];
    for (int j = 0; j < x; j++)
        dis[0][j + 1] = dis[0][j] + h[0][j];
    for (int i = 0; i < y; i++)
        for (int j = 0; j < x; j++)
            _________; // 在此处填写代码
    return dis[y][x];
}
```

- A. `dis[i][j] = min(dis[i - 1][j] + v[i - 1][j], dis[i][j - 1] + h[i][j - 1]);`
- B. `dis[i][j] = min(dis[i - 1][j] + h[i - 1][j], dis[i][j - 1] + v[i][j - 1]);`
- C. `dis[i + 1][j + 1] = min(dis[i][j + 1] + v[i][j + 1], dis[i + 1][j] + h[i + 1][j]);`

☐ **D.** `dis[i + 1][j + 1] = min(dis[i][j + 1] + h[i][j + 1], dis[i + 1][j] + v[i + 1][j]);`

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 C++语言非常强大，可以用来求解方程的解。例如，如果变量 `x` 为 `double` 类型的变量，则执行语句 `x * 2 - 4 = 0;` 后，变量 `x` 的值会变为 `2.0` 。

第 2 题 一个袋子中有3个完全相同的红色小球、2个完全相同的蓝色小球。每次从中取出1个，且不放回袋子，这样进行3次后，将取出的小球依次排列，则可能的颜色顺序有7种。

第 3 题 杨辉三角，是二项式系数的一种三角形排列，在中国南宋数学家杨辉1261年所著的《详解九章算法》一书中出现，是中国数学史上的一项伟大成就。

第 4 题 N个顶点的有向完全图（不带自环）有$N \times (N-1)/2$条边。

第 5 题 如果待查找的元素确定，只要哈希表的大小不小于查找元素的个数，就一定存在不会产生冲突的哈希函数。

第 6 题 动态规划算法的时间复杂度一般为：必要状态的数量，乘以计算一次状态转移方程的时间复杂度。

第 7 题 已知 `int` 类型的变量 `a` 、 `b` 和 `h` 中分别存储着一个梯形的顶边长、底边长和高，则这个梯形的面积可以通过表达式 `(a + b) * h / 2` 求得。

第 8 题 判断图是否连通只能用广度优先搜索算法实现。

第 9 题 在N个元素的二叉排序树中查找一个元素，最好情况的时间复杂度是$O(\log N)$。

第 10 题 给定 `double` 类型的变量 `x` ，且其值大于等于0，我们可以通过二分法求出$\sqrt{x}$的近似值。

3　编程题（每题 25 分，共 50 分）

3.1　编程题 1

- **试题名称**：奖品分配
- **时间限制**：1.0 s
- **内存限制**：128.0 MB

3.1.1　问题描述

班上有 N 名同学，学号从 0 到 $N-1$。有 M 种奖品要分给这些同学，其中，第 i 种奖品总共有 a_i 个（$i = 0, 1, \ldots, M-1$）。巧合的是，奖品的数量不多不少，每位同学都可以恰好分到一个奖品，且最后剩余的奖品不超过 1 个（即：$N \leq a_0 + a_1 + \cdots + a_{M-1} \leq N+1$）。

现在，请你求出每个班级礼物分配的方案数，所谓方案，指的是为每位同学都分配一个种类的奖品。只要有一位同学获得了不同种类的奖品，即视为不同的方案。方便起见，你只需要输出方案数对 10^9+7 取模后的结果即可。

共有 T 个班级都面临着奖品分配的问题，你需要依次为他们解答。

3.1.2 输入描述

第一行一个整数 T，表示班级数量。

接下来 T 行，每行若干用单个空格隔开的正整数。首先是两个正整数 N, M，接着是 M 个正整数 $a_0, a_1, \ldots, a_{M-1}$。保证 $N \leq a_0 + a_1 + \cdots + a_{M-1} \leq N + 1$。

3.1.3 输出描述

输出 T 行，每行一个整数，表示该班级分配奖品的方案数对 $10^9 + 7$ 取模的结果。

3.1.4 特别提醒

在常规程序中，输入、输出时提供提示是好习惯。但在本场考试中，由于系统限定，请不要在输入、输出中附带任何提示信息。

3.1.5 样例输入 1

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

3.1.6 样例输出 1

```
3
4
20
```

3.1.7 样例解释 1

对于第 1 个班级，学号为 $0, 1, 2$ 的同学可以依次分别获得奖品 $0, 1, 1$，也可以依次分别获得奖品 $1, 0, 1$，也可以依次分别获得奖品 $1, 1, 0$，因此共有 3 种方案。

对于第 2 个班级，学号为 $0, 1, 2$ 的同学可以依次分别获得奖品 $0, 1, 1$，也可以依次分别获得奖品 $1, 0, 1$，也可以依次分别获得奖品 $1, 1, 0$，也可以依次分别获得奖品 $1, 1, 1$，因此共有 4 种方案。

对于第 3 个班级，可以把编号为 1 的奖品分配给 5 名同学中的任意一名，共有 5 种方案；再把编号为 2 的奖品分配给剩余 4 名同学中的任意一名，共有 4 种方案；最后给剩余 3 名同学自然获得 0 号奖品。因此，方案数为 $5 \times 4 = 20$。

3.1.8 样例输入 2

```
5
100 1 100
100 1 101
20 2 12 8
123 4 80 20 21 3
999 5 101 234 499 66 99
```

3.1.9 样例输出 2

```
1
1
125970
895031741
307187590
```

3.1.10 数据规模

对于30%的测试点，保证 $N \leq 10$。

对于另外30%的测试点，保证 $M = 2$。

对于所有测试点，保证 $N \leq 1,000$；保证 $T \leq 1,000$；保证 $M \leq 1,001$。

3.1.11 参考程序

```cpp
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <algorithm>
#include <string>
#include <map>
#include <iostream>
#include <cmath>
#include <vector>
using namespace std;
const int N = 1005;
const int mod = 1e9 + 7;
int C[N + 5][N + 5], a[N + 5];
void add(int &a, int b) {
    a += b; if(a >= mod) a -= mod;
}
void init() {
    C[0][0] = 1;
    for(int i = 1; i <= N; i ++) {
        C[i][0] = C[i][i] = 1;
        for(int j = 1; j < i; j ++)
            C[i][j] = C[i - 1][j - 1], add(C[i][j], C[i - 1][j]);
    }
}
int main() {
    // freopen("data/10.in", "r", stdin);
    init();
    int T;
    scanf("%d", &T);
    while(T --) {
        int n, m, sum = 0;
        scanf("%d%d", &n, &m);
        for(int i = 1; i <= m; i ++)
            scanf("%d", &a[i]), sum += a[i];

        int ans = 1;
        for(int i = 1; i <= m; i ++) {
            ans = 1ll * ans * C[sum][a[i]] % mod;
```

```
            sum -= a[i];
        }
        cout << ans << endl;
    }
    return 0;
}
```

3.2 编程题 2

- **试题名称**：大量的工作沟通
- **时间限制**：2.0 s
- **内存限制**：128.0 MB

3.2.1 问题描述

某公司有 N 名员工，编号从 0 至 $N-1$。其中，除了 0 号员工是老板，其余每名员工都有一个直接领导。我们假设编号为 i 的员工的直接领导是 f_i。

该公司有严格的管理制度，每位员工只能受到本人或直接领导或间接领导的管理。具体来说，规定员工 x 可以管理员工 y，当且仅当 $x=y$，或 $x=f_y$，或 x 可以管理 f_y。特别地，0 号员工老板只能自我管理，无法由其他任何员工管理。

现在，有一些同事要开展合作，他们希望找到一位同事来主持这场合作，这位同事必须能够管理参与合作的所有同事。如果有多名满足这一条件的员工，他们希望找到编号最大的员工。你能帮帮他们吗?

3.2.2 输入描述

第一行一个整数 N，表示员工的数量。

第二行 $N-1$ 个用空格隔开的正整数，依次为 $f_1, f_2, \ldots, f_{N-1}$。

第三行一个整数 Q，表示共有 Q 场合作需要安排。

接下来 Q 行，每行描述一场合作：开头是一个整数 m（$2 \le m \le N$），表示参与本次合作的员工数量；接着是 m 个整数，依次表示参与本次合作的员工编号（保证编号合法且不重复）。

保证公司结构合法，即不存在任意一名员工，其本人是自己的直接或间接领导。

3.2.3 输出描述

输出 Q 行，每行一个整数，依次为每场合作的主持人选。

3.2.4 特别提醒

在常规程序中，输入、输出时提供提示是好习惯。但在本场考试中，由于系统限定，请不要在输入、输出中附带任何提示信息。

3.2.5 样例输入 1

```
5
0 0 2 2
3
2 3 4
3 2 3 4
2 1 4
```

3.2.6 样例输出 1

```
2
2
0
```

3.2.7 样例解释 1

对于第一场合作，员工 $3,4$ 有共同领导 2，可以主持合作。

对于第二场合作，员工 2 本人即可以管理所有参与者。

对于第三场合作，只有 0 号老板才能管理所有员工。

3.2.8 样例输入 2

```
7
0 1 0 2 1 2
5
2 4 6
2 4 5
3 4 5 6
4 2 4 5 6
2 3 4
```

3.2.9 样例输出 2

```
2
1
1
1
0
```

3.2.10 数据规模

对于25%的测试点，保证 $N \leq 50$。

对于50%的测试点，保证 $N \leq 300$。

对于所有测试点，保证 $3 \leq N \leq 10^5$；保证 $Q \leq 100$，保证 $m \leq 10^4$。

3.2.11 参考程序

```
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <algorithm>
#include <string>
#include <map>
#include <iostream>
#include <cmath>
#include <vector>
#include <queue>
using namespace std;
const int N = 100005;
int fa[N], sz[N], dep[N], son[N], tp[N], mxId[N];
int cnt, fir[N], tar[N], nxt[N];
```

```
void link(int a, int b) {
    tar[++ cnt] = b, nxt[cnt] = fir[a], fir[a] = cnt;
}
void dfs(int x, int mxid) {
    int Mx = 0; sz[x] = 1;
    mxId[x] = max(x, mxid);
    for (int i = fir[x]; i; i = nxt[i]) {
        dep[tar[i]] = dep[x] + 1;
        dfs(tar[i], mxId[x]);
        sz[x] += sz[tar[i]];
        if (Mx < sz[tar[i]])
            Mx = sz[son[x] = tar[i]];
    }
}
void gettp(int x) {
    tp[x] = x;
    if (son[fa[x]] == x)
        tp[x] = tp[fa[x]];
    for (int i = fir[x]; i; i = nxt[i])
        gettp(tar[i]);
}
int lca(int x,int y){
    while (tp[x] != tp[y])
        dep[tp[x]] > dep[tp[y]] ? x = fa[tp[x]] : y = fa[tp[y]];
    return dep[x] < dep[y] ? x : y;
}

int main() {
    int n;
    scanf("%d", &n);
    for(int i = 2; i <= n; i ++)
        scanf("%d", &fa[i]), link(++ fa[i], i);
    dfs(1, 1), gettp(1);

    int q;
    scanf("%d", &q);
    while(q --) {
        int m, x, y;
        scanf("%d%d", &m, &x), x ++;
        for(int i = 2; i <= m; i ++)
            scanf("%d", &y), x = lca(x, ++ y);
        cout << mxId[x] - 1 << endl;
    }
    return 0;
}
```