洛阳市 2026 年中小学信息学奥林匹克竞赛初赛

（满分：100 分 考试时间：90 分钟）

第一部分：单项选择题（共 15 题，每题 2 分，共计 30 分）

1. 一个智能空调的规则是：如果 “温度高于 28 度” 或 “湿度高于 80%”，则自动开启制冷。今天空调启动了制冷。那么以下哪种情况一定不正确？（ ）

A. 温度 26 度，湿度 85%

B. 温度 29 度，湿度 75%

C. 温度 27 度，湿度 78%

D. 温度 30 度，湿度 90%

2. 智能音箱在你说话后，需要依次完成以下步骤才能回答你，正确的顺序应该是:（ ）① 理解你话里的意思② 播放回答的语音③ 思考并生成回答的内容④ 接收你的语音

A. ④ → ① → ③ → ②

B. ④ → ③ → ① → ②

C. ① → ④ → ③ → ②

D. ③ → ① → ④ → ②

3. 回文子串指的是一个字符串中，正着读和反着读都一样的连续字符序列。S = "CABCBAABCABCBAB"，其最长回文子串的长度是（ ）

A.5 B.6 C.7 D.8

4. 对于给定的数字 11223344，如果每次取连续的一段（长度至少为 1）构成一个新的数字，问能构成多少个不同的数字（ ）。

A. 14 B. 27 C. 32 D. 36

5. 以下排序算法中，时间复杂度在最坏情况下仍为 O(n log n)的是（ ）

A. 冒泡排序 B. 快速排序 C. 归并排序 D. 插入排序

6. 某商队运送茶叶，每箱装 15 斤。现有 100 斤茶叶，计算能装满多少箱以及剩余多少斤。以下代码片段正确的是？

```
1. int boxs = 100 / 15
2. int remainder = 100 % 15 ;  //   %为取余数符号
```

执行后，boxs 和 remainder 的值分别是？

A. 6, 10　　B. 7, 5　　C. 6, 5　　D. 7, 10

7. 有 12 个人在玩游戏，他们围成一个圈，给定一个字符串 LLLLRRLRRRLL 代表每个人的 “攻击” 方向（L 向左，R 向右）。一个合法的 “攻击” 应满足以下二者之一：

①若只有 a 攻击 b，则 b 必须攻击 a。

②若 a 和 c 同时攻击 b，或 a 和 c 都不攻击 b，则 b 可以任意攻击 a 和 c 中的一个。

你每次可以进行一次操作，使字符串中的一个字符从 L 变为 R，或从 R 变为 L。最少需要操作（ ）次使得字符串所代表的 “攻击” 状态合法。

A. 1　　B. 2　　C. 3　　D. 4

8. 陶陶和乐乐最近在练习跳跃技巧，跳跃规则如下：起点位于数轴的点 0，如果想要到达点 x，一开始可以先跳一单位，然后每次跳跃的长度都比上一次多一单位。每次跳跃可以选择向左或向右。现在陶陶想跳到坐标为 56 的位置，乐乐想跳到坐标为 61 的位置，他们两个人分别最少跳跃（ ）次才能到达自己的目标点。（ ）

A. 10,11　　B. 11,12　　C. 12,13　　D. 11,13

9. 同学们玩猜城市游戏，地图上有标号 A、B、C、D、E。五人每人只答对一半：

甲说：B 是北京，E 是天津

乙说：B 是湖北，D 是重庆

丙说：C 是湖北，D 是吉林

丁说：A 是重庆，E 是吉林

戊说：B 是天津，C 是北京

正确选项是(　　)

A. A 是重庆，B 是天津，C 是湖北，D 是北京，E 是吉林

B. A 是湖北，B 是天津，C 是重庆，D 是吉林，E 是北京

C. A 是重庆，B 是湖北，C 是北京，D 是天津，E 是吉林

D. A 是重庆，B 是湖北，C 是北京，D 是吉林，E 是天津

10. 斐波那契数列定义为: (F(0) = 0, F(1) = 1, F(n) = F(n - 1) + F(n - 2)

（n>=2）。用 Si 表示前 i 项和，则 Sn 可以表示为（　　）

A. F(n+1)

B. F(n+2)

C. F(n+1)-1

D. F(n+2)-1

11. 在贸易数据统计中，初始计数器 count = 5。执行语句

```
int x = ++count;
int y = count++;
int total = x + y;
```

后，变量 total 和 count 的值分别是？（　　）

A. total=12, count=7　　B. total=13, count=7

C. total=11, count=6　　D. total=12, count=6

12. 洛阳厂区的集装箱排列成矩形阵列。为了测试系统，程序员编写了一个类似“九九乘法表”的逻辑来生成坐标编号。执行以下代码后，当 i=2 时，内层循环一共执行了多少次？（　　）

```
for(int i = 1; i <= 5; i++) {
for(int j = 1; j <= i; j++) {
    cout << i << "*" << j << " ";
    }
    cout << endl;
}
```

A. 1　　B. 2　　C. 5　　D. 10

13. 以下代码执行后，变量 y 的值为（　　）

```
int x = 1, y = 0;
while (x <= 1024) {
    x = x * 2;
    y = y + 1;
}
```

A. 9　　B. 10　　C. 11　　D. 12

14. 有如下递归函数：

```
int f(int n) {
    if (n <= 1) return 1;
    return n * f(n - 2);
}
```

调用 f(7)的返回值为（　　）

A. 7　　B. 105　　C. 210　　D. 5040

15.以下哪个是计算机科学领域的最高奖项？（　　）

A. 诺贝尔奖　　B. 图灵奖　　C. 菲尔兹奖　　D. 普利策奖

第二部分：阅读程序题（判断题 2 分/题，选择题 3 分/题，共计 40 分）

说明：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，判断题每题 2 分，选择题每题 3 分。

阅读程序（一）求斐波那契数列第 n 项的值。

```
1.  #include <iostream>
2.  using namespace std;
3.
4.  int f(int n) {
5.      if (n <= 0) return 0;
6.      if (n == 1) return 1;
7.      return f(n - 1) + f(n - 2);
8.  }
9.
10. int main() {
11.     int n;
12.     cin >> n;
13.     cout << f(n) << endl;
14.     return 0;
15. }
```

假设输入的 n 是不超过 30 的正整数。

判断题（每题 2 分）

16.1.该程序的第 n 个数的结果由 n-1 的结果和 n-2 的结果相加得到。（　　）

16.2. 该程序 f(1) = 1。（　　）

16.3. 该程序 f(2) = 2。（　　）

选择题（每题 3 分）

16.4.该递归算法中，递归树的最大深度（即调用栈的最大深度）与 n 的关系是（　　）

A. 最大深度约为 n/2

B. 最大深度等于 n

C. 最大深度约为 2n

D. 最大深度约为 2^n

16.5.以下关于该程序的优化方案中，正确的是（　　）

A. 改用递推（循环）实现，可将时间复杂度从 $O(2^n)$ 降为 $O(n)$，空间复杂度降为 $O(1)$

B. 改用记忆化搜索，可将时间复杂度降为 $O(n)$，但空间复杂度会增至 $O(2^n)$

C. 该递归程序已经是最优实现，无需优化

D. 无论采用何种优化，斐波那契数列的计算时间复杂度下限为 $O(2^n)$

阅读程序（二）

```
#include <iostream>
#include <stack>
using namespace std;
bool visit[30];
bool flag[30];
int main() {
    string s;
    cin >> s;
    for(int i = 0; i < s.size(); i++)
    {
        if ((visit[s[i] - 'a'])
        {
            flag[s[i] - 'a'] = true;
        }
        visit[s[i] - 'a'] = true;
    }

    for(int i = 0; i < s.size(); i++)
    {
        if ((!flag[s[i] - 'a'])
        {
            cout << s[i];
            return 0;
        }
    }
    cout << "No";
```

```
    return 0;
}
```

假设输入的字符串长度不超过 1000。

判断题（每题 2 分）

17.1.程序在遇到第一个只出现一次的字符时，这个字符的 flag 的结果为 false。（　　）

17.2.当输入为“112233”时，程序输出“No”。（　　）

17.3.如果输入“113220”，程序将输出“0”。（　　）

选择题（每题 3 分）

17.4.当输入为{[(]时，程序的输出为（　　）

A. (

B. {

C. [

D. No

17.5.若字符串的长度为 n，则该算法的时间复杂度为（　　）

A. O(1)

B. O(n)

C. $O(n^2)$

D. O(log n)

17.6.若将输入改为((()))[][]{}{{}}，程序的输出为（　　）

A. (

B. {

C. [

D. No

阅读程序（三）计算 n 的二进制表示中 1 的个数。

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

const int MOD = 10753;

```

```
int calc(int n) {
        int x = n;
        while (x) {
            if (x & 1) cnt++;
            x >>= 1;
        }
    return cnt%MOD;
}

int main() {
    int n;
    cin >> n;
    cout << calc(n) << endl;
    return 0;
}
```

假设输入的 n 是不超过 10^6 的正整数。

判断题（每题 2 分）

18.1.函数 calc(n)的结果对 10753 取模。（ ）

18.2.当输入 $n=8$ 时，程序输出结果为 1。（ ）

选择题（每题 3 分）

18.3.该程序的时间复杂度为（ ）

A. O(n)

B. O(n log n)

C. $O(n^2)$

D. O(log n)

18.4.当 $n=2^k-1$（k 为正整数）时，程序输出的结果（忽略取模）可表示为（ ）

A. k

B. k − 1

C. k +1

D. 1

18.5.当 $n=2^k$ （k 为正整数）时，程序输出的结果（忽略取模）可表示为（ ）

A. k

B. k − 1

C. k +1

D. 1

第三部分：完善程序题（单选题，每题 3 分，共计 30 分）

程序（一）

（求最大公约数——欧几里得算法）输入两个正整数 a 和 b，输出它们的最大公约数。

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

int gcd(int a, int b) {
    while (①) {
        int r = a % b;
        a = b;
        b = r;
    }
    return ②;
}

int main() {
    int a, b;
    cin >> a >> b;
    cout << gcd(a, b) << endl;
    return 0;
}
```

19.①处应填（　　）

A. a % b != 0　　B. b != 0　　C. a > b　　D. a != 0

20.②处应填（　　）

A. a　　B. b　　C. a % b　　D. r

21.该算法的时间复杂度为（　　）

A. O(n)　　B. O(log n)　　C. $O(n^2)$　　D. O(2^n)

程序（二）

（求无向图最短路径） 给定一个包含 n 个顶点、m 条边的无向图，所有边的长度均为 1。求从顶点 s 到顶点 t 的最短路径长度。若无法到达，则输出 -1。

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

const int MAXN = 100005;
vector<int> G[MAXN];       // 邻接表存储图
int dist[MAXN];            // dist[i] 记录从起点 s 到 i 的最短距离，-1 表示未访问

int bfs(int s, int t, int n) {
    // 初始化距离数组
    for (int i = 1; i <= n; i++) dist[i] = ①;

    queue<int> q;
    dist[s] = 0;
    q.push(s);

    while (②) {
        int u = q.front();
        q.pop();
        if (u == t) return dist[u];

        // 遍历 u 的所有邻接点
        for (int v : G[u]) {
            if (dist[v] == -1) {
                dist[v] = ③;
                q.push(v);
            }
        }
    }
    return -1;
}

int main() {
    int n, m, s, t;
    cin >> n >> m >> s >> t;
    for (int i = 0; i < m; i++) {
```

```
        int u, v;
        cin >> u >> v;
        G[u].push_back(v);
        G[v].push_back(u);
    }
    cout << bfs(s, t, n) << endl;
    return 0;
}
```

22. ①处应填（　　）

A. 0

B. -1

C. INF

D. s

23. ②处应填（　　）

A. q.size() > 0

B. q.front() != -1

C. !q.empty()

D. u != t

24. ③处应填（　　）

A. dist[v] + 1

B. dist[u] + 1

C. dist[u]

D. dist[v] + dist[u]

程序（三）

（前缀和与差分应用）你是学校的管理员，有很多班级向你提交“教室使用申请单”，你需要处理 n 天和 m 份订单。第 i 天有 r[i] 个空余教室。每份订单包含三个正整数 d, s, t，表示从第 s 天到第 t 天每天需借 d 个教室。按订单顺序处理，若某订单导致任一天教室不足，则输出 -1 和该订单编号；若全部满足，则输出 0。

```
#include<iostream>
#include<cstdio>
#include<cstring>
#define MAXN 1000005
using namespace std;
struct cf{
```

```
    int x,l,r;
};
cf f[MAXN];
int n,m,a[MAXN],b[MAXN],c[MAXN];

// 判断执行前 x 个操作后，所有元素是否都不超过上限
bool chafen(int x)
{
    memset(b,0,sizeof(b));
    for(int i=1;i<=x;i++)
    {
        ①;
        b[f[i].r+1]-=f[i].x;
    }
    for(int i=1;i<=n;i++)
    {
        ②;
        if(c[i]>a[i])
            return 0;
    }
    return 1;
}

int main()
{
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;i++)
        scanf("%d",&a[i]);
    for(int i=1;i<=m;i++)
        scanf("%d%d%d",&f[i].x,&f[i].l,&f[i].r);

    if(chafen(m))
        printf("0");
    else
    {
        int l=1,r=m;
        while(l<r)
        {
            ③;
            if(chafen(mid))
                l=mid+1;
            else
                r=mid;
        }
```

```
51.         printf("-1\n%d",l);
52.     }
53.     return 0;
54. }
```

25. ①处应填（　　）

A. b[f[i].l] -= f[i].x

B. b[f[i].l] += f[i].x

C. b[f[i].l-1] += f[i].x

D. b[f[i].r] += f[i].x

26. ②处应填入的代码是（　　）

A. c[i] = b[i-1] + c[i]

B. c[i] = b[i] + c[i-1]

C. c[i] = c[i-1] + b[i-1]

D. c[i] = b[i] - c[i-1]

27. ③处应填入的代码是（　　）

A. int mid = (l + r + 1) >> 1

B. int mid = (l + r) >> 1

C. int mid = l + (r - l + 1) / 2

D. int mid = (l + r) / 2 + 1

28.下列关于 chafen 函数的说法中，错误的是（　　）

A. memset(b, 0, sizeof(b))的作用是每次调用时清空差分数组，保证多次调用之间的独立性

B. 当传入参数 x=0 时，chafen 函数一定会返回 true

C. 如果某个操作的 r 等于 n，那么 b[f[i].r+1] -= f[i].x 这行代码会导致数组越界错误

D. 前缀和计算必须从 i=1 开始，若从 i=0 开始会导致结果错误