全国青少年信息素养大赛-2026-初中组-算法-Cpp-星火-复赛-模拟题-01-答案与解析

1	2	3	4	5	6	7	8	9	10
B	A	A	B	A	ACD	ACD	ACD	ABC	ABC

一、单选题

（1）边区兵工厂的弹药传送带是一个环形流水线，一头装填弹药、一头取走成品。我们用 `front` 指向当前可取弹药的队头位置， `rear` 指向队尾元素的下一个位置（即下一个可装填弹药的空闲位置），传送带最大容量为 `maxSize` 。判断传送带已满、不能再装填的条件是（ ）。

A. `rear == front`
B. `(rear + 1) % maxSize == front`
C. `(rear - 1 + maxSize) % maxSize == front`
D. `(rear - 1) == front`

【正确答案】B

【题目解析】
核心考点：
循环队列的队满判断——牺牲一个位置区分队空与队满。

关键辨析：
`A` 选项 `rear == front` 是队空条件，不是队满。循环队列规定当 `(rear + 1) % maxSize == front` 时，即队尾再向后走一步就追上队头，此时判为队满。

易错提醒：
`C` 和 `D` 为迷惑选项，混淆了下标前进与后退方向。牢记公式：队空 `front == rear` ，队满 `(rear + 1) % maxSize == front` 。

（2）报务员截获多份敌军电台频率，同一频率可能被多次截获。为去重并按频率值有序排列，使用 `set<int>` 存储已确认的敌军频率。阅读下面代码，输出结果是（ ）。

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

int main() {
    set<int> freq;
    freq.insert(45);
    freq.insert(12);
    freq.insert(45);
    freq.insert(78);
    freq.insert(12);
    cout << freq.size() << " ";
    cout << *freq.begin() << " ";
    cout << *freq.rbegin();
    return 0;
}
```

A. `3 12 78`

B. `5 45 12`

C. `3 45 78`

D. `5 12 78`

【正确答案】A

【题目解析】

核心考点：

考察 `set` 的两大核心特性——自动去重 + 自动按 key 排序。

执行追踪：

`insert(45)` → `{45}`；`insert(12)` → `{12, 45}`；`insert(45)` → 重复，忽略，仍为 `{12, 45}`；`insert(78)` → `{12, 45, 78}`；`insert(12)` → 重复，忽略。最终 `size()` = 3，`*begin()` 返回最小元素 12，`*rbegin()` 返回最大元素 78。

易错提醒：

`set` 自动去重——相同元素多次 `insert()` 只保留一个。`begin()` 指向最小元素，`rbegin()` 指向最大元素。不要去数 `insert()` 调用次数来判断 `size()`，重复插入不增加元素数量。

（3）某作战部队的花名册采用双向链表管理，以便快速增删调动人员。现需将一名已调离的战士从链表中移除。下面代码实现该删除操作，横线处应填写（ ）。

```
void deleteNode(DoublyListNode*& head, int value) {
    DoublyListNode* current = head;
    while (current != nullptr && current->val != value)
        current = current->next;
    if (current != nullptr) {
        if (current->prev != nullptr) {
            _____________________________ // 在此处填入代码
        } else {
            head = current->next;
        }
        if (current->next != nullptr)
            current->next->prev = current->prev;
        delete current;
    }
}
```

A. `current->prev->next = current->next;`

B. `current->next = current->prev;`

C. `delete current->next;`

D. `current->prev = current->next;`

【正确答案】A

【题目解析】

核心考点：

考察双向链表删除操作的指针调整——需要同时处理前驱和后继。

执行追踪：

删除双向链表中的节点需要：(1) 若被删节点不是头节点，将其前驱的 `next` 指向其后继：`current->prev->next = current->next`；(2) 若被删节点不是尾节点，将其后继的 `prev` 指向其前驱：`current->next->prev = current->prev`。代码中横线处处理的是第一项（头节点的 `prev` 为 `nullptr`）。

易错提醒：

双向链表删除涉及两个指针的调整（前驱的 next 和后继的 prev），缺一不可。代码先处理前驱（横线处），再处理后继（后面已给出），两者配合完成删除。

（4）多路部队的战果分别统计后，需合并为一份完整的战报，同时要求相等的歼敌数保持原有上报次序。归并排序正适合这一需求。下面关于归并排序的描述，正确的是（ ）。

A. 归并排序是一个不稳定的排序算法

B. 归并排序的时间复杂度在最优、最差和平均情况下都是 $O(n \log n)$

C. 归并排序需要额外的 $O(1)$ 空间

D. 对于输入数组 $\{12, 11, 13, 5, 6, 7\}$，代码输出结果为 $7\ 6\ 5\ 13\ 12\ 11$

【正确答案】B

【题目解析】

核心考点：

考察归并排序的基本性质——稳定性、时间/空间复杂度及排序结果。

选项分析：

A 错误，归并排序是稳定的排序算法——合并左右子数组时，相等元素优先取左侧，保持原有相对顺序。

B 正确，归并排序采用稳定二分策略，任何时候复杂度都是 $O(n \log n)$ 。

C 错误，归并排序合并阶段需要临时数组暂存结果，额外空间为 $O(n)$ ，不是 $O(1)$ 。

D 错误，归并排序将数组按升序排列，输出应为 $5\ 6\ 7\ 11\ 12\ 13$ 。

易错提醒：

归并排序是三大稳定排序之一（冒泡、插入、归并）。其 $O(n)$ 额外空间是主要代价，但在链表排序中可降至 $O(1)$ 。

（5）后勤处将各连队的补给物资按连队编号登记造册，同一连队多次登记的以最新数据覆盖旧数据。这种"编号→物资"的映射关系恰好用 `map` 实现。下面代码完成登记与查询，横线处应填入（ ）。

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

int main() {
    map<int, int> supply;        // 连队编号 → 物资数量
    supply[3] = 150;
    supply[1] = 200;
    supply[3] = 180;             // 3号连队物资更新，覆盖旧值
    supply[2] = 120;

    cout << _________________;   // 输出登记过的不同连队个数
    supply.erase(1);             // 1号连队已完成补给，移除记录

    for (auto it = supply.begin(); it != supply.end(); it++)
        cout << it->first << ":" << it->second << " ";
    return 0;
}
```

A. `supply.size()`

B. `supply.length()`

C. `supply.count()`

D. `supply.capacity()`

【正确答案】A

【题目解析】

核心考点：

考察 `map` 的基本操作——`size()`、键唯一性、`erase()`、遍历输出。

执行追踪：

虽然执行了 4 次 `[]` 赋值，但 `supply[3]` 出现了两次，第二次将值从 150 更新为 180（覆盖旧值），因此 `size()` 返回 3（三个不同的键：$1, 2, 3$）。`erase(1)` 移除键 1，遍历输出为 `2:120 3:180`（`map` 自动按键升序排列）。

易错提醒：

`map` 的 `[]` 运算符若键不存在则插入新元素，若键已存在则覆盖旧值——不会增加元素数量。`count(key)` 返回某键的出现次数（对 `map` 只能为 0 或 1），不能用于求总元素个数。`map` 没有 `length()` 或 `capacity()` 方法。

二、多选题

（6）边区后勤处在设计物资数据库时，需要为不同种类的数据分配合适的存储类型——人数用整数、重量用长整数、标签用字符、状态用布尔值。关于 C++ 中常见数据类型及其大小的说法，正确的有（ ）。

A. `int` 类型占 4 字节，`long long` 类型占 8 字节
B. `char` 类型占 2 字节
C. `double` 类型占 8 字节
D. `bool` 类型占 1 字节

【正确答案】ACD

【题目解析】
核心考点：
考察 C++ 基本数据类型的字节大小记忆。

选项分析：
A 正确，`int` 占 4 字节（32 位），`long long` 占 8 字节（64 位）。
B 错误，`char` 占 1 字节（8 位），存储 ASCII 码值。
C 正确，`double` 占 8 字节（64 位），双精度浮点数。
D 正确，`bool` 占 1 字节，虽然理论上 1 位就够了，但 C++ 中最小可寻址单元是 1 字节。

易错提醒：
`char` 是 1 字节不是 2 字节；不同平台下数据类型大小可能略有差异，但以上为通用标准。

（7）行军计划的编排经常用到不同的循环结构：已知行军天数的用固定次数的循环，条件未知的用判断式循环，至少执行一次任务的用先执行后判断的循环。关于 C++ 中三种循环结构的特征，以下说法正确的有（ ）。

A. `for` 循环适合循环次数已知的场景
B. `while` 循环在条件判断前至少执行一次循环体
C. `do-while` 循环在条件判断前至少执行一次循环体
D. `for` 循环的初始化部分中声明的变量，其作用域仅限于该 `for` 语句

【正确答案】ACD

【题目解析】
核心考点：
考察三种循环结构的适用场景与关键区别。

选项分析：
A 正确，`for` 将初始化、条件、步进集中书写，最适合"已知循环次数"的场景。

B 错误，"至少执行一次"是 `do-while` 的特性，`while` 是先判断后执行的。
C 正确，`do-while` 先执行循环体再判断条件。
D 正确，`for (int i = 0; i < n; i++)` 中 `i` 的作用域仅限于 `for` 语句内，循环结束后 `i` 不可访问。

易错提醒：
选择循环结构的原则——已知次数用 `for`，未知次数用 `while`，至少一次用 `do-while`。

（8）电台通信中每个字母和数字都有固定的编码值（ASCII 码）。报务员需要熟练进行字符与编码之间的转换，例如将数字字符转为实际数值、判断字母大小写等。关于字符与 ASCII 码，以下说法正确的有（ ）。

A. 字符 `'0'` 的 ASCII 码是 48
B. 大写字母的 ASCII 码比对应小写字母大 32
C. 将数字字符转换为对应的整数值，可以用 `ch - '0'`
D. 判断字符 `ch` 是否为小写字母可以用 `ch >= 'a' && ch <= 'z'`

【正确答案】ACD

【题目解析】
核心考点：
考察字符与 ASCII 码的基本关系。

选项分析：
A 正确，`'0'` 的 ASCII 码为 48，`'A'` 为 65，`'a'` 为 97。
B 错误，小写字母的 ASCII 码比对应大写字母大 32（即 `'a' - 'A' = 32`），选项说法正好反了。
C 正确，`'5' - '0' = 53 - 48 = 5`，这是字符转数字的标准方法。
D 正确，利用 ASCII 码的连续性判断字符范围。

易错提醒：
务必牢记 `'a'` 比 `'A'` 大 32，即小写字母 ASCII 码更大。

（9）各军分区上报的战功数据需按歼敌数排序，但同属一个分区的部队在排序后应保持内部原有先后顺序，这就对排序算法的"稳定性"提出了要求。关于排序算法的稳定性，以下说法正确的有（ ）。

A. 冒泡排序是稳定的排序算法，因为相等元素不会发生交换
B. 插入排序是稳定的排序算法

C. 选择排序是不稳定的排序算法，因为在交换最小值时可能改变相等元素的相对顺序
D. 快速排序是稳定的排序算法

【正确答案】ABC

【题目解析】
核心考点：
考察常见排序算法的稳定性判断。

选项分析：
A 正确，冒泡排序在相等元素时不交换，保证稳定性。
B 正确，插入排序将元素插入到已排序序列的合适位置时，遇到相等元素会放在其后，不会跨越。
C 正确，选择排序如 `[5_1, 5_2, 3]` → 选 3 与 5_1 交换得 `[3, 5_2, 5_1]`，两个 5 相对顺序改变。
D 错误，快速排序在分区过程中可能跨越相等元素，是不稳定的。

易错提醒：
稳定排序：冒泡、插入、归并。不稳定排序：选择、快排、堆排序。

（10）前线阵地上的弹药箱堆放遵循"后放上的先取用"原则，这正是栈（Stack）的数据结构思想。在作战行动中，每一次推进和回撤的路线记录也如同栈的压入和弹出。关于栈，以下说法正确的有（ ）。

A. 栈是一种后进先出（LIFO）的数据结构
B. 栈的基本操作包括 `push`（入栈）和 `pop`（出栈）
C. 函数调用过程中，局部变量和返回地址通常存储在系统栈中
D. 在 C++ STL 中，`stack` 的 `top()` 操作不仅返回栈顶元素，还会将该元素移除

【正确答案】ABC

【题目解析】
核心考点：
全面考察栈的基本概念、操作和应用。

选项分析：
A 正确，栈是 LIFO 结构。
B 正确，`push` 和 `pop` 是栈的两个核心操作。
C 正确，函数调用时参数、返回地址和局部变量压入系统调用栈，返回时弹出——这正是"栈"名称的由来。
D 错误，STL 中 `top()` 只返回栈顶元素的值（不删除），`pop()` 才移除栈顶元素但不返回其值。两者需要配合使用。

易错提醒：
C++ STL `stack` 的 `pop()` 返回类型为 `void` （不返回值），这是为了异常安全。必须先 `top()` 获取值，再 `pop()` 移除。

三、编程题

（1）Y3722 即时战功评功

描述

某次战役后，各参战部队的战功数据陆续上报至指挥部。指挥部按上报顺序逐一评定即时立功分数线：设立功比例为 $w\%$ ，已上报 p 份战功数据时，计划评功人数为 $\max(1, \lfloor p \times w\% \rfloor)$ 。如有战功数值相同，则所有并列者均获评功，实际评功人数可能超过计划数。
请你模拟这一实时评功过程：每收到一份新战功数据，立即输出当前的评功分数线（即当前排名前 $w\%$ 的最低战功值）。

输入描述

第一行有两个整数 n, w （ $1 \leq n \leq 100{,}000$ ， $1 \leq w \leq 99$ ），分别代表战功数据总数与立功比例。第二行有 n 个整数，依次代表逐一上报的战功值（ $0 \leq$ 战功值 ≤ 600 ）。

输出描述

输出一行，包含 n 个非负整数，依次代表每份战功数据上报后的即时评功分数线。相邻两个整数间用一个空格分隔。

输入样例 1

```
10 60
200 300 400 500 600 600 0 300 200 100
```

输出样例 1

```
200 300 400 400 400 500 400 400 300 300
```

（2）Y3723 情报暗号核对

描述

地下党接头时使用暗号字符串 A 和 B （长度不超过 50 ，只含小写字母）。为避免因口音或抄写误差导致误判，规定：若 若 A 与 B 完全相同，或 A 通过删除一个字符、插入一个字符或修改

一个字符能够变成 B ，则视为暗号匹配，输出 `"similar"` ；否则输出 `"not similar"` 。共有 T 组暗号待核对（ $T \leq 100$ ）。

输入描述

第一行一个整数 T ，接下来 T 行每行两个字符串 A 和 B 。

输出描述

对于每组数据，输出 `"similar"` 或 `"not similar"` ，每组占一行。

输入样例 1

```
5
apple applee
apple appe
apple bpple
applee bpple
apple apple
```

输出样例 1

```
similar
similar
similar
not similar
similar
```

（3）Y3724 军需预算规划

描述

八路军某师后勤部记录了接下来 N （ $1 \leq N \leq 100{,}000$ ）天的每日军需开销。

输入描述

第一行包含两个整数 N, M ，用单个空格隔开。

输出描述

输出一个整数，即最大预算周期开销的最小值。

输入样例 1

```
7 5
100
400
300
100
500
101
400
```

输出样例 1

```
500
```

（4）Y3725 密电频率筛查

描述

报务员截获了 n 份敌军电报，按截获时间顺序编号为 $1 \sim n$ ，每份电报有一个频率值。为研判敌军是否在同一频率上频繁更换加密方式，参谋需要找出频率值相同的电报中，截获序号间隔最近的那一对。如果有多对间隔相同，输出其中左端点在数组中位置最靠前（即左端点下标最小）的那对对应的频率值；若所有频率值均互不相同，输出 `No` 。

输入描述

第一行一个整数 n （ $1 \leq n \leq 10^5$ ）。第二行 n 个整数，依次表示每份电报的频率值（绝对值不超过 10^9 ）。

输出描述

输出一个整数，即符合条件的最小频率值；若无相同频率值，输出 `No` 。

输入样例 1

```
7
1 3 2 1 2 3 1
```

输出样例 1

```
2
```