

C++编程等级认证五级 2023 年 09 月理论 60 分钟

【单选题】(每题 2 分)

1、近年来，线上授课变得普遍，很多有助于改善教学效果的设备也逐渐流行，其中包括比较常用的手写板，那么它属于哪类设备？
()

A、输入 B、输出 C、控制 D、记录

2、如果 a 和 b 均为 int 类型的变量，且 b 的值不为 0，那么下列能正确判断“a 是 b 的 3 倍”的表达式是？()

A、(a >> 3 == b) B、(a - b) % 3 == 0 C、(a / b == 3) D、(a == 3 * b)

3、如果变量 a 和 b 分别为 double 类型和 int 类型，则表达式 (a = 6, b = 3 * (7 + 8) / 2, b += a) 的计算结果为？()

A、6 B、21 C、28 D、不确定

4、有关下面 C++ 代码说法错误的是？()

// sumA()和 sumB()用于求从 1 到 N 之和

```
#include <iostream>
```

```
using namespace std;
```

```
int sumA(int n) {
```

```
    int sum = 0;
```

```
    for (int i = 1; i < n + 1; i++)
```

```
        sum += i;
```

```
    return sum;
```

```
}
```

```
int sumB(int n) {
```

```
    if (n == 1)
```

```
        return 1;
```

```
    else
```

```
        return n + sumB(n - 1);
```

```
}
```

```
int main() {
```

```
    int n = 0;
```

```
    cin >> n;
```

```
    cout << sumA(n) << " " << sumB(n) << endl;
```

```
    return 0;
```

```
}
```

A、sumA() 用循环方式求从 1 到 N 之和，sumB() 用递归方式求从 1 到 N 之和。

B、默认情况下，如果输入正整数 1000，能实现求从 1 到 1000 之和。

C、默认情况下，如果输入正整数 100000，能实现求从 1 到 100000 之和。

D、一般说来，sumA() 的效率高于 sumB()。

5、下面 C++ 代码以递归方式实现字符串反序，横线处应填上代码是？()

//字符串反序

```
#include <iostream>
```

```
#include <string>
```

```
using namespace std;
```

```
string sReverse(string sIn) {
```

```
    if (sIn.length() <= 1) {
```

```
        return sIn;
```

```
    } else {
```

```
        return _____ // 此处填写代码
```

```
}
```

```
int main() {
```

```

string sIn;
cin >> sIn;
cout << sReverse(sIn) << endl;
return 0;
}
A、sIn[sIn.length() - 1] + sReverse(sIn.substr(0, sIn.length() - 1));
B、sIn[0] + sReverse(sIn.substr(1, sIn.length() - 1));
C、sReverse(sIn.substr(0, sIn.length() - 1)) + sIn[sIn.length() - 1];
D、sReverse(sIn.substr(1, sIn.length() - 1)) + sIn[sIn.length() - 1];

```

6、印度古老的汉诺塔传说：创世时有三根金刚柱，其中一柱从下往上按照大小顺序摆着 64 片黄金圆盘，当圆盘逐一从一柱借助另外一柱全部移动到另外一柱时，宇宙毁灭。

移动规则：在小圆盘上不能放大圆盘，在三根柱子之间一次只能移动一个圆盘。下面的 C++ 代码以递归方式实现汉诺塔，横线处应填入代码是？（ ）

```

#include <iostream>
using namespace std;
// 递归实现汉诺塔，将 N 个圆盘从 A 通过 B 移动 C
// 圆盘从底到顶，半径必须从大到小
void Hanoi(string A, string B, string C, int N) {
    if (N == 1){
        cout << A << " -> " << C << endl;
    } else {
        Hanoi(A, C, B, N - 1);
        cout << A << " -> " << C << endl;
        _____; // 此处填写代码
    }
}
}
int main() {
    Hanoi("甲", "乙", "丙", 3);
    return 0;
}

```

- A、Hanoi(B, C, A, N - 2)
- B、Hanoi(B, A, C, N - 1)
- C、Hanoi(A, B, C, N - 2)
- D、Hanoi(C, B, A, N - 1)

7、根据下面 C++ 代码的注释，两个横线处应分别填入？（ ）

```

#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
bool isOdd(int N) {
    return N % 2 == 1;
}
bool compare(int a, int b) {
    if (a % 2 == 0 && b % 2 == 1)
        return true;
    return false;
}
int main() {

```

```

vector<int> lstA; // lstA 是一个整型向量
for (int i = 1; i < 100; i++)
    lstA.push_back(i);
// 对 lstA 成员按比较函数执行结果排序
sort(lstA.begin(), lstA.end(), _____); // 此处填写代码 1
vector<int> lstB;
for (int i = 0; i < lstA.size(); i++) // lstB 成员全为奇数
    if ( _____ ) // 此处填写代码 2
        lstB.push_back(lstA[i]);
cout << "lstA : ";
for (int i = 0; i < lstA.size(); i++)
    cout << lstA[i] << " ";
cout << endl;
cout << "lstB : ";
for (int i = 0; i < lstB.size(); i++)
    cout << lstB[i] << " ";
cout << endl;
return 0;
}

```

- A、compare 和 isOdd(lstA[i])
- B、compare(x1, y1) 和 isOdd
- C、compare 和 isOdd
- D、compare(x1, y1) 和 isOdd(lstA[i])

8、有关下面代码正确的是？（ ）

// 在 C++ 语言中，可以通过函数指针的形式，将一个函数作为另一个函数的参数。
 // 具体来说：bool checkNum(bool(*Fx)(int), int N); 声明了一个函数，
 // 其第一个参数是函数指针类型，指向一个接收一个 int 参数且返回值为 bool 的函数。

```

#include <iostream>
using namespace std;
bool isEven(int N) {
    return N % 2 == 0;
}
bool checkNum(bool (*Fx)(int), int N){
    return Fx(N);
}
int main() {
    cout << checkNum(isEven, 10) << endl;
    return 0;
}

```

- A、checkNum() 函数定义错误。
- B、将 isEven 作为 checkNum() 参数将导致错误。
- C、执行后将输出 1。
- D、运行时触发异常。

9、有关下面 C++ 代码正确的是？（ ）

```

#include <iostream>
using namespace std;
bool isOdd(int N) {
    return N % 2 == 1;
}

```

```

}
int Square(int N){
    return N * N;
}
bool checkNum(bool (*Fx)(int), int x){
    return Fx(x);
}
int main() {
    cout << checkNum(isOdd, 10) << endl; // 输出行 A
    cout << checkNum(Square, 10) << endl; // 输出行 B
    return 0;
}

```

- A、checkNum() 函数定义错误。
 B、输出行 A 的语句将导致编译错误。
 C、输出行 B 的语句将导致编译错误。
 D、该代码没有编译错误。

10、下面代码执行后的输出是？ ()

```

#include <iostream>
using namespace std;
int jumpFloor(int N) {
    cout << N << "#";
    if(N == 1 || N == 2) {
        return N;
    } else {
        return jumpFloor(N - 1) + jumpFloor(N - 2);
    }
}
int main() {
    cout << jumpFloor(4) << endl;
    return 0;
}

```

- A、4#3#2#2#4 B、4#3#2#2#1#5 C、4#3#2#1#2#4 D、4#3#2#1#2#5

11、下面代码中的 isPrimeA() 和 isPrimeB() 都用于判断参数 N 是否素数，有关其时间复杂度的正确说法是？ ()

```

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

bool isPrimeA(int N) {
    if(N < 2)
        return false;
    for (int i = 2; i < N; i++)
        if (N % i == 0)
            return false;
    return true;
}

bool isPrimeB(int N) {
    if (N < 2)

```

```

        return false;
int endNum = int(sqrt(N));
for (int i = 2; i <= endNum; i++)
    if (N % i == 0)
        return false;
return true;
}

int main() {
    cout << boolalpha;
    cout << isPrimeA(13) << " " << isPrimeB(13) << endl;
    return 0;
}

```

- A、isPrimeA() 的最坏时间复杂度是 $O(N)$ ，isPrimeB() 的最坏时间复杂度是 $O(\log N)$ ，isPrimeB() 优于 isPrimeA()。
- B、isPrimeA() 的最坏时间复杂度是 $O(N)$ ，isPrimeB() 的最坏时间复杂度是 $O(N^{\frac{1}{2}})$ ，isPrimeB() 优于 isPrimeA()。
- C、isPrimeA() 的最坏时间复杂度是 $O(N^{\frac{1}{2}})$ ，isPrimeB() 的最坏时间复杂度是 $O(N)$ ，isPrimeA() 优于 isPrimeB()。
- D、isPrimeA() 的最坏时间复杂度是 $O(\log N)$ ，isPrimeB() 的最坏时间复杂度是 $O(N)$ ，isPrimeA() 优于 isPrimeB()。

12、下面代码用于归并排序，其中 merge() 函数被调用次数为？()

```

#include <iostream>
using namespace std;

void mergeSort(int * listData, int start, int end);
void merge(int * listData, int start, int middle, int end);
void mergeSort(int* listData, int start, int end) {
    if (start >= end)
        return;
    int middle = (start + end) / 2;
    mergeSort(listData, start, middle);
    mergeSort(listData, middle + 1, end);
    merge(listData, start, middle, end);
}

void merge(int * listData, int start, int middle, int end) {
    int leftSize = middle - start + 1;
    int rightSize = end - middle;
    int * left = new int[leftSize];
    int * right = new int[rightSize];
    for (int i = 0; i < leftSize; i++)
        left[i] = listData[start + i];
    for (int j = 0; j < rightSize; j++)
        right[j] = listData[middle + 1 + j];
    int i = 0, j = 0, k = start;
    while (i < leftSize && j < rightSize) {
        if (left[i] <= right[j]) {
            listData[k] = left[i];
            i++;
        } else {

```

```

        listData[k] = right[j];
        j++;
    }
    k++;
}

while (i < leftSize) {
    listData[k] = left[i];
    i++;
    k++;
}
while (j < rightSize) {
    listData[k] = right[j];
    j++;
    k++;
}
delete[] left;
delete[] right;
}

int main() {
    int lstA[] = {1, 3, 2, 7, 11, 5, 3};
    int size = sizeof(lstA) / sizeof(lstA[0]);
    mergeSort(lstA, 0, size - 1); // 对 lstA 执行归并排序
    for (int i = 0; i < size; i++)
        cout << lstA[i] << " ";
    cout << endl;
    return 0;
}

```

A、0 B、1 C、6 D、7

13、在上题的归并排序算法中，mergeSort(listData, start, middle); 和 mergeSort(listData, middle + 1, end); 涉及到的算法为？（ ）

A、搜索算法 B、分治算法 C、贪心算法 D、递推算法

14、归并排序算法的基本思想是？（ ）

- A、将数组分成两个子数组，分别排序后再合并。
- B、随机选择一个元素作为枢轴，将数组划分为两个部分。
- C、从数组的最后一个元素开始，依次与前一个元素比较并交换位置。
- D、比较相邻的两个元素，如果顺序错误就交换位置。

15、有关下面代码的说法正确的是？（ ）

```

#include <iostream>

class Node {
public:
    int Value;
    Node * Next;
    Node(int Val, Node *Nxt = nullptr) {
        Value = Val;
        Next = Nxt;
    }
}

```

```
};
int main() {
    Node * firstNode = new Node(10);
    firstNode->Next = new Node(100);
    firstNode->Next->Next = new Node(111, firstNode);
    return 0;
}
```

A、上述代码构成单向链表。 B、上述代码构成双向链表。 C、上述代码构成循环链表。 D、上述代码构成指针链表。

【判断题】(每题 2 分)

- 16、TCP/IP 的传输层的两个不同的协议分别是 UDP 和 TCP。()
- 17、在特殊情况下流程图中可以出现三角框和圆形框。()
- 18、找出自然数 N 以内的所有质数，常用算法有埃氏筛法和线性筛法，其中埃氏筛法效率更高。()
- 19、在 C++ 中，可以使用二分法查找链表中的元素。()
- 20、在 C++ 中，通过恰当的实现，可以将链表首尾相接，形成循环链表。()
- 21、贪心算法的解可能不是最优解。()
- 22、一般说来，冒泡排序算法优于归并排序。()
- 23、C++ 语言中的 qsort 库函数是不稳定排序。()
- 24、质数的判定和筛法的目的并不相同，质数判定旨在判断特定的正整数是否为质数，而质数筛法意在筛选出范围内的所有质数。()
- 25、下面的 C++ 代码执行后将输出 0 5 1 6 2 3 4。()

```
#include <iostream>
#include <algorithm>
using namespace std;
bool compareModulo5(int a, int b) {
    return a % 5 < b % 5;
}
int main() {
    int lst[7];
    for (int i = 0; i < 7; i++)
        lst[i] = i;
    //对序列所有元素按 compareModulo5 结果排序
    sort(lst, lst + 7, compareModulo5);
    for (int i = 0; i < 7; i++)
        cout << lst[i] << " ";
    cout << endl;
    return 0;
}
```

【编程题】

26、(本题 25 分) 因数分解

时间限制：1.0 s

内存限制：128.0 MB

【问题描述】

每个正整数都可以分解成素数的乘积，例如： $6 = 2 \times 3$ 、 $20 = 2^2 \times 5$ 。

现在，给定一个正整数 N，请按要求输出它的因数分解式。

【输入描述】

输入第一行，包含一个正整数 N。约定 $2 \leq N \leq 10^{12}$ 。

【输出描述】

输出一行，为 N 的因数分解式。要求按质因数由小到大排列，乘号用星号*表示，且左右各空一格。当且仅当一个素数出现多次时，将它们合并为指数形式，用上箭头^表示，且左右不空格。

【样例输入 1】

6

【样例输出 1】

2 * 3

【样例输入 2】

20

【样例输出 2】

2^2 * 5

【样例输入 3】

23

【样例输出 3】

23

27、(本题 25 分) 巧夺大奖

时间限制：1.0 s

内存限制：128.0 MB

【问题描述】

小明参加了一个巧夺大奖的游戏节目。主持人宣布了游戏规则：

- (1) 游戏分为 n 个时间段，参加者每个时间段可以选择一个小游戏。
 - (2) 游戏中共有 n 个小游戏可供选择。
 - (3) 每个小游戏有规定的时限和奖励。对于第 i 个小游戏，参加者必须在第 T_i 个时间段结束前完成才能得到奖励 R_i 。
- 小明发现，这些小游戏都很简单，不管选择哪个小游戏，他都能在一个时间段内完成。关键在于，如何安排每个时间段分别选择哪个小游戏，才能使得总奖励最高？

【输入描述】

输入第一行，包含一个正整数 n 。既是游戏时间段的个数，也是小游戏的个数。约定 $1 \leq n \leq 500$ 。

输入第二行，包含 n 个正整数。第 i 个正整数为 T_i ，即第 i 个小游戏的完成期限。约定 $1 \leq T_i \leq n$ 。

输入第三行，包含 n 个正整数。第 i 个正整数为 R_i ，即第 i 个小游戏的完成奖励。约定 $1 \leq R_i \leq 1000$ 。

【输出描述】

输出一行，包含一个正整数 C ，为最高可获得的奖励。

【样例输入 1】

7

4 2 4 3 1 4 6

70 60 50 40 30 20 10

【样例输出 1】

230

【样例解释 1】

7 个时间段可分别安排完成第 4、2、3、1、6、7、5 个小游戏，其中第 4、2、3、1、7 个小游戏在期限内完成。因此，可以获得总计 $40+60+50+70+10=230$ 的奖励。