

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

C++ 五级

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
<u>答案</u>															

第 1 题 下面 C++ 代码用于求斐波那契数列，该数列第 1、2 项为 1，以后各项均是前两项之和。下面有关说法错误的是()。

```
1 int fiboA(int N)
2 {
3     if (N == 1 || N == 2)
4         return 1;
5     return fiboA(N - 1) + fiboA(N - 2);
6 }
7 int fiboB(int N)
8 {
9     if (N == 1 || N == 2)
10        return 1;
11     int last2 = 1, last1 = 1;
12     int nowVal = 0;
13     for (int i = 2; i < N; i++)
14     {
15         nowVal = last1 + last2;
16         last2 = last1;
17         last1 = nowVal;
18     }
19     return nowVal;
20 }
```

- A. `fiboA()` 用递归方式，`fiboB()` 循环方式
- B. `fiboA()` 更加符合斐波那契数列的数学定义，直观易于理解，而 `fiboB()` 需要将数学定义转换为计算机程序实现
- C. `fiboA()` 不仅仅更加符合数学定义，直观易于理解，且因代码量较少执行效率更高
- D. `fiboB()` 虽然代码量有所增加，但其执行效率更高

第 2 题 下面 C++ 代码以递归方式实现合并排序，并假设 `merge (int T[], int R[], int s, int m, int t)` 函数将有序（同样排序规则）的 `T[s..m]` 和 `T[m+1..t]` 归并到 `R[s..t]` 中。横线处应填上代码是()。

```
1 void mergeSort(int SList[], int TList[], int s, int t, int len)
2 {
3     if (s == t){
4         TList[s] = SList[s];
5         return;
6     }
7     int *T2 = new int[len]; // 保存中间结果
8     int m = (s + t) / 2;
9     _____;
10    merge(T2, SList, s, m, t);
11    delete T2;
12    return ;
13 }
```

- A. `mergeSort(SList, T2, s, m, len), mergeSort(SList, T2, m, t, len)`

- B. mergeSort(SList, T2, s, m-1, len), mergeSort(SList, T2, m+1, t, len)
- C. mergeSort(SList, T2, s, m, len), mergeSort(SList, T2, m+1, t, len)
- D. mergeSort(SList, T2, s, m-1, len), mergeSort(SList, T2, m-1, t, len)

第3题 阅读下面的C++代码，执行后其输出是()。

```

1 int stepCount = 0;
2 int fracA(int N)
3 {
4     stepCount += 1;
5     cout << stepCount << " -> ";
6     int rtn = 1;
7     for (int i = 1; i <= N; i++)
8         rtn *= i;
9     return rtn;
10 }
11 int fracB(int N)
12 {
13     stepCount += 1;
14     cout << stepCount << " -> ";
15     if (N == 1)
16         return 1;
17     return N * fracB(N - 1);
18 }
19 int main()
20 {
21     cout << fracA(5);
22     cout << "<====>";
23     cout << fracB(5);
24     return 0;
25 }
```

- A. 1->120<====>2->120
- B. 1->120<====>1->120
- C. 1->120<====>1->2->3->4->5->120
- D. 1->120<====>2->3->4->5->6->120

第4题 下面的C++用于对 lstA 排序，使得偶数在前奇数在后，横线处应填入()。

```

1 bool isEven(int N)
2 {
3     return N % 2 == 0;
4 }
5
6 void swap(int &a, int &b)
7 {
8     int t;
9     t=a, a=b, b=t;
10    return;
11 }
12
13 void sortA(int lstA[], int n)
14 {
15     int i, j, t;
16     for (i = n-1; i > 0; i--)
17         for(j = 0; j < i; j++)
18             if (
19                 swap(lstA[j], lstA[j+1]));
20
21     return;
22 }
```

- A. isEven(lstA[j]) && !isEven(lstA[j+1])
- B. !isEven(lstA[j]) && isEven(lstA[j+1])
- C. lstA[j] > lstA[j+1]

D. $\text{lstA}[j] < \text{lstA}[j+1]$

第5题 下面的C++代码用于将字符串保存到带头节点的双向链表中，并对重复的串计数，然后将最新访问的串的节点放在链头便于查找。横线处应填入代码是（ ）。

```
1 typedef struct Node{
2     string str;
3     int ref;
4     struct Node *next, *prev;
5 }Node;
6 Node * Insert(Node *pHead, string s)
7 {
8     Node *p = pHead->next;
9     Node *q;
10    while(p){
11        if(p->str == s) {
12            p->ref++;
13            p->next->prev = p->prev;
14            p->prev->next = p->next;
15            break;
16        }
17        p=p->next;
18    }
19    if(!p) {
20        p = new Node;
21        p->str = s;
22        p->ref=0;
23        p->next = p->prev = NULL;
24    }
25
26    pHead->next = p, p->prev = pHead;
27    return pHead;
28 }
```

A. **if**(pHead) {p->next = pHead->next, pHead->next->prev = p;}

B. **if**(pHead->next) {p->next = pHead->next, pHead->next->prev = p;}

C. p->next = pHead->next, pHead->next->prev = p;

D. 触发异常，不能对空指针进行操作。

第6题 有关下面C++代码说法正确的是（ ）。

```
1 int rc;
2 int foo(int x, int y)
3 {
4     int r;
5     if(y == 0)
6         r = x;
7     else{
8         r = foo(y, x % y);
9         rc++;
10    }
11    return r;
12 }
```

A. 如果 x 小于10, rc 值也不会超过20

B. foo 可能无限递归

C. foo 可以求出 x 和 y 的最大公共质因子

D. foo 能够求出 x 和 y 的最小公倍数

第7题 下面的C++代码实现对list的快速排序，有关说法，错误的是（ ）。

```

1 vector<int> operator +(vector<int>lA, vector<int>lB)
2 {
3     vector<int>lst;
4
5     for (int i = 1; i < lA.size(); i++)
6         lst.push_back(lA[i]);
7     for (int i = 1; i < lB.size(); i++)
8         lst.push_back(lB[i]);
9
10    return lst;
11 }
12
13 vector<int>qSort(vector<int>lst)
14 {
15     if (lst.size() < 2)
16         return lst;
17     int pivot = lst[0];
18     vector<int>less, greater;
19     for (int i = 1; i < lst.size(); i++)
20         if (lst[i] <= pivot) less.push_back(lst[i]);
21         else greater.push_back(lst[i]);
22     for (int i = 1; i < lst.size(); i++)
23         if (lst[i] <= pivot) less.push_back(lst[i]);
24         else greater.push_back(lst[i]);
25
26     return _____;
27 }

```

- A. qSort(less) + qSort(greater) + (vector<int>)pivot
- B. (vector<int>)pivot + (qSort(less) + qSort(greater))
- C. (qSort(less) + (vector<int>)pivot + qSort(greater))
- D. qSort(less) + pivot + qSort(greater)

第8题 下面C++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（ ）。

```

1 bool isPrimeA(int N)
2 {
3     if (N < 2)
4         return false;
5     for (int i = 2; i <= N / 2 ; i++)
6         if (N % i == 0)
7             return false;
8     return true;
9 }
10 bool isPrimeB(int N)
11 {
12     if (N < 2)
13         return false;
14     for (int i = 2; i <= sqrt(N) ; i++)
15         if (N % i == 0)
16             return false;
17     return true;
18 }

```

- A. isPrimeA() 的最坏时间复杂度是 $O(\frac{N}{2})$, isPrimeB() 的最坏时间复杂度是 $O(\log N)$, isPrimeA() 优于 isPrimeB()
- B. isPrimeA() 的最坏时间复杂度是 $O(\frac{N}{2})$, isPrimeB() 的最坏时间复杂度是 $O(N^{\frac{1}{2}})$, isPrimeB() 绝大多数情况下优于 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()

第9题 下面C++代码用于有序 list 的二分查找，有关说法错误的是（ ）。

```

1 int _binarySearch(vector<int>lst, int Low, int High,int Target)
2 {
3     if (Low > High)
4         return -1;
5     int Mid = (Low + High) / 2;
6     if (Target == lst[Mid])
7         return Mid;
8     else if (Target < lst[Mid])
9         return _binarySearch(lst, Low, Mid - 1, Target);
10    else
11        return _binarySearch(lst, Mid + 1, High, Target);
12 }
13 int bSearch(vector<int>lst, int Val)
14 {
15     return _binarySearch(lst, 0, lst.size(), Val);
16 }

```

- A. 代码采用二分法实现有序 list 的查找
- B. 代码采用分治算法实现有序 list 的查找
- C. 代码采用递归方式实现有序 list 的查找
- D. 代码采用动态规划算法实现有序 list 的查找

第 10 题 在上题的 `_binarySearch` 算法中，如果 `lst` 中有 N 个元素，其时间复杂度是（ ）。

- A. $O(N)$
- B. $O(\log N)$
- C. $O(N \log N)$
- D. $O(N^2)$

第 11 题 下面的C++代码使用数组模拟整数加法，可以处理超出大整数范围的加法运算。横线处应填入代码是（ ）。

```

1 vector<int> operator +(vector<int> a, vector<int> b)
2 {
3     vector<int> c;
4     int t = 0;
5
6     for(int i = 0; i < a.size() || i < b.size(); i ++ )
7     {
8         if(i < a.size()) t = t + a[i];
9         if(i < b.size()) t = t + b[i];
10    }
11    _____
12
13    if(t) c.push_back(t);
14
15    return c;
16 }

```

- A. `c.push_back(t % 10), t = t % 10;`
- B. `c.push_back(t / 10), t = t % 10;`
- C. `c.push_back(t / 10), t = t / 10;`
- D. `c.push_back(t % 10), t = t / 10;`

第 12 题 有关下面C++代码的说法正确的是（ ）。

```

1 class Node
2 {
3 public:
4     int Value;
5     Node* Prev;
6     Node* Next;
7     Node(int Val, Node* Prv = NULL, Node* Nxt = NULL);
8 };
9
10 Node::Node(int Val, Node* Prv, Node* Nxt)
11 {
12     this->Value = Val;
13     this->Prev = Prv;
14     this->Next = Nxt;
15 }
16
17 int main()
18 {
19     Node firstNode = Node(10);
20     firstNode.Next = new Node(100, &firstNode);
21     firstNode.Next->Next = new Node(111, firstNode.Next);
22 }

```

- 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

第 13 题 通讯卫星在通信网络系统中主要起到（）的作用。

- A. 信息过滤
- B. 信号中继
- C. 避免攻击
- D. 数据加密

第 14 题 小杨想编写一个判断任意输入的整数N是否为素数的程序，下面哪个方法不合适？（）

- A. 埃氏筛法
- B. 线性筛法
- C. 二分答案
- D. 枚举法

第 15 题 下面的排序算法都要处理多趟数据，哪种排序算法不能保证在下一趟处理时从待处理数据中选出最大或最小的数据？（）

- A. 选择排序
- B. 快速排序
- C. 堆排序
- D. 冒泡排序

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 归并排序的时间复杂度是 $O(N \log N)$ 。 ()

第 2 题 小杨在生日聚会时拿一块 $H \times W$ 的巧克力招待来的 K 个小朋友，保证每位小朋友至少能获得一块相同大小的巧克力。那么小杨想分出来最大边长的巧克力可以使用二分法。 ()

第 3 题 以下 C++ 代码能以递归方式实现斐波那契数列，该数列第 1、2 项为 1，以后各项均是前两项之和。 ()

```
1 int Fibo(int N)
2 {
3     if (N == 1 || N == 2)
4         return 1;
5     else
6     {
7         int m = fiboA(N - 1);
8         int n = fiboB(N - 2);
9         return m + n;
10    }
11 }
```

第 4 题 贪心算法可以达到局部最优，但可能不是全局最优解。 ()

第 5 题 小杨设计了一个拆数程序，它能够将任意的非质数自然数 N 转换成若干个质数的乘积，这个程序是可以设计出来的。 ()

第 6 题 插入排序有时比快速排序时间复杂度更低。 ()

第 7 题 下面的 C++ 代码能实现十进制正整数 N 转换为八进制并输出。 ()

```
1 char s[10];
2 int main()
3 {
4     int N;
5     cin >> N;
6     string rst = "";
7     while (N != 0)
8     {
9         s[0] = N % 8 + '0';
10        rst += string(s);
11        N /= 8;
12    }
13    cout << rst << endl;
14
15    return 0;
16 }
```

第 8 题 对数组 `int arr[] = {2, 6, 3, 5, 4, 8, 1, 0, 9, 10}` 执行 `sort(arr, arr+10)`，则执行后 `arr` 中的数据调整为 `{0, 1, 2, 3, 4, 5, 6, 8, 9, 10}`。 ()

第 9 题 小杨想写一个程序来算出正整数 N 有多少个因数，经过思考他写出了一个重复没有超过 $N/2$ 次的循环就能够算出来了。 ()

第 10 题 同样的整数序列分别保存在单链表和双向链中，这两种链表上的简单冒泡排序的复杂度相同。 ()