2023 CCF⾮专业级别软件能⼒认证第⼀轮

（CSP - J1）⼊⻔级C++语⾔试题

**认证时间：2023年9⽉16⽇09:30～11:30**

⼀、 单项选择题（共15题，每题2分，共计30分：每题有且仅有⼀个正确选项）

1. 在C++中，下⾯哪个关键字⽤于声明⼀个变量，其值不能被修改？（ ）。

1. unsigned
2. const
3. static
4. mutable

2. ⼋进制数123456708 和076543218的和为（ ）。

A. 222222218

B. 211111118

C. 221111118

D. 222222118

1. 阅读下述代码，请问修改data的value成员以存储3.14，正确的⽅式是（ ）。
2. union Data{
3. int num;
4. float value;
5. char symbol; 5 };

6 union Data data;

1. data.value = 3.14;
2. value.data = 3.14;
3. data->value = 3.14;
4. value->data = 3.14;
5. 假设有⼀个链表的节点定义如下：
6. struct Node {
7. int data;
8. Node\* next; 4 };

现在有⼀个指向链表头部的指针：Node\* head。如果想要在链表中插⼊⼀个新节点，其成员data的值为42，并使新节点成为链表的第⼀个节点，下⾯哪个操作是正确的？（ ）

1. Node\* newNode = new Node; newNode->data = 42; newNode->next = head; head = newNode;
2. Node\* newNode = new Node; head->data = 42; newNode->next = head; head = newNode;
3. Node\* newNode = new Node; newNode->data = 42; head->next = newNode;
4. Node\* newNode = new Node; newNode->data = 42; newNode->next = head;
5. 根节点的⾼度为1，⼀根拥有2023个节点的三叉树⾼度⾄少为（ ）。
6. 6
7. 7
8. 8
9. 9
10. ⼩明在某⼀天中依次有七个空闲时间段，他想要选出⾄少⼀个空闲时间段来练习唱歌，但他希望任意两个练习的时间段之间都有⾄少两个空闲的时间段让他休息，则⼩明⼀共有（ ）种选择时间段的⽅案。
11. 31
12. 18
13. 21
14. 33
15. 以下关于⾼精度运算的说法错误的是（ ）。
16. ⾼精度计算主要是⽤来处理⼤整数或需要保留多位⼩数的运算。
17. ⼤整数除以⼩整数的处理的步骤可以是，将被除数和除数对⻬，从左到右逐位尝试将除数乘以某个数，通过减法得到新的被除数，并累加商。
18. ⾼精度乘法的运算时间只与参与运算的两个整数中⻓度较⻓者的位数有关。
19. ⾼精度加法运算的关键在于逐位相加并处理进位。

8. 后缀表达式“6 2 3 + - 3 8 2 / + \* 2 ^ 3 +”对应的中缀表达式是（ ）

A. ((6 - (2 + 3)) \* (3 + 8 / 2)) ^ 2 + 3

B. 6 - 2 + 3 \* 3 + 8 / 2 ^ 2 + 3

C. (6 - (2 + 3)) \* ((3 + 8 / 2) ^ 2) + 3

D. 6 - ((2 + 3) \* (3 + 8 / 2)) ^ 2 + 3

9. 数1010102和1668的和为（ ）。

A. 101100002

B. 2368

C. 15810

D. A016

1. 假设有⼀组字符{a,b,c,d,e,f}，对应的频率分别为5%，9%，12%，13%，16%，45%。请问以下哪个选项是字符a,b,c,d,e,f分别对应的⼀组哈夫曼编码？（ ）

A. 1111，1110，101，100，110，0

B. 1010，1001，1000，011，010，00

C. 000，001，010，011，10，11

D. 1010，1011，110，111，00，01

1. 给定⼀棵⼆叉树，其前序遍历结果为：ABDECFG，中序遍历结果为：DEBACFG。请问这棵树的正确后序遍历结果是什么？（ ）
2. EDBFGCA
3. EDBGCFA
4. DEBGFCA
5. DBEGFCA
6. 考虑⼀个有向⽆环图，该图包括4条有向边：(1,2)，(1,3)，(2,4)，和(3,4)。以下哪个选项是这个有向⽆环图的

⼀个有效的拓扑排序？（ ）

A. 4，2，3，1

B. 1，2，3，4

C. 1，2，4，3

D. 2，1，3，4

1. 在计算机中，以下哪个选项描述的数据存储容量最⼩？（ ）
2. 字节（byte）
3. ⽐特（bit）
4. 字（word）
5. 千字节（kilobyte）
6. ⼀个班级有10个男⽣和12个⼥⽣。如果要选出⼀个3⼈的⼩组，并且⼩组中必须⾄少包含1个⼥⽣，那么有多少种可能的组合？（ ）

A. 1420

B. 1770

C. 1540

D. 2200

1. 以下哪个不是操作系统？（ ）
2. Linux
3. Windows
4. Android
5. HTML

⼆、 阅读程序（程序输⼊不超过数组成字符串定义的范围：判断题正确填√，错误填×；除特殊说明外，判断题1.5分，选择题3分，共计40分）

(1）

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

#include <iostream>

#include <cmath> using namespace std;

double f(double a, double b, double c) { double s = (a + b + c) / 2;

return sqrt(s \* (s - a) \* (s - b) \* (s - c));

}

int main() {

cout.flags(ios::fixed); cout.precision(4);

int a, b, c;

cin >> a >> b >> c;

cout << f(a, b, c) << endl; return 0;

}

假设输⼊的所有数都为不超过1000的正整数，完成下⾯的判断题和单选题：判断题:

16. （2分）当输⼊为“2 2 2”时，输出为“1.7321”（ ）

1. （2分）将第7⾏中的"（s-b）（*s-c*）*"*改为*"*（*s-c*）（s-b）"不会影响程序运⾏的结果（ ）
2. （2分）程序总是输出四位⼩数（ ）单选题
3. 当输⼊为“3 4 5”时，输出为（ ）

A. "6.0000" B. "12.0000" C. "24.0000" D. "30.0000"

20. 当输⼊为“5 12 13”时，输出为（ ）

（2）

* 1. #include <iostream>
  2. #include <vector>
  3. #include <algorithm>

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 4 | using namespace std; | | | |  |
| 5 | int f(string x, string y) { | | | |
| 6 | int m = x.size(); | | | |
| 7 | int n = y.size(); | | | |
| 8 | vector<vector<int>> v(m + 1, vector<int>(n + 1, | | | | 0)); |
| 9 | for (int i = 1; i <= m; i++) { | | | |  |
| 10 | for (int j = 1; j <= n; j++) { | | | |  |
| 11 | if (x[i - 1] == y[j - 1]) { | | | |  |
| 12 | v[i][j] = v[i - 1][j - 1] + 1; | | | |  |
| 13 | } else { | | | |  |
| 14 | v[i][j] = max(v[i - | 1][j], | v[i][j | - | 1]); |
| 15 | } |  | | | |
| 16 | } |
| 17 | } |
| 18 | return v[m][n]; |
| 19 | } |
| 20 |  |
| 21 | bool g(string x, string y) { |
| 22 | if (x.size() != y.size()) { |
| 23 | return false; |
| 24 | } |
| 25 | return f(x + x, y) == y.size(); |
| 26 | } |
| 27 |  |
| 28 | int main() { |
| 29 | string x, y; |
| 30 | cin >> x >> y; |
| 31 | cout << g(x, y) << endl; |
| 32 | return 0; |
| 33 | } |

判断题

1. f函数的返回值⼩于等于min（n,m）。（ ）
2. f函数的返回值等于两个输⼊字符串的最⻓公共⼦串的⻓度。（ ）
3. 当输⼊两个完全相同的字符串时，g函数的返回值总是true（ ）单选题
4. 将第19⾏中的“v[m][n]”替换为“v[n][m]”，那么该程序（ ）
5. 当输⼊为“csppsc spsccp”时，输出为：（ ）

（3）

1 #include <iostream>

int main() {

int n; cin >> n;

cout << solve2(solve1(n)) << " " << solve1((solve2(n))) << endl; return 0;

}

int solve2(int n) {

int sum = 0;

for (int i = 1; i <= sqrt(n); i++) { if (n % i == 0) {

if (n / i == i) { sum += i \* i;

} else {

sum += i \* i + (n / i) \* (n / i);

}

}

}

return sum;

}

#include <cmath>

using namespace std; int solve1(int n) {

return n \* n;

}

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

假设输⼊的n是绝对值不超过1000的整数，完成下⾯的判断题和单选题。判断题

1. 如果输⼊的n为正整数，solve2函数的作⽤是计算n所有的因⼦的平⽅和（ ）
2. 第13~14⾏的作⽤是避免n的平⽅根因⼦i（或n/i）进⼊第16⾏⽽被计算两次（ ）
3. 如果输⼊的n为质数，solve2（n）的返回值为n2+1（ ）单选题
4. （4分）如果输⼊的n为质数p的平⽅，那么solve2（n）的返回值为（ ）

A. p2+p+1 B. n2+n+1 C. n2+1 D. p4+2p2+1

1. 当输⼊为正整数时，第⼀项减去第⼆项的差值⼀定（ ）

A. ⼤于0 B. ⼤于等于0且不⼀定⼤于0 C. ⼩于0 D. ⼩于等于0且不⼀定⼩于0

1. 当输⼊为“5”时，输出为（ ）

A. "651.625" B. "650.729" C. "651.676" D. "652.625"

三、完善程序(单选题，每⼩题3分，共计 3 分)

(1)(寻找被移除的元素)问题:原有⻓度为 n+1公差为1等升数列，将数列输到程序的数组时移除了⼀个元素，导致⻓度为 n 的开序数组可能不再连续，除⾮被移除的是第⼀个或最后之个元素。需要在数组不连续时，找出被移除的元素。试补全程序。

|  |  |  |
| --- | --- | --- |
| 1 | #include<iostream> |  |
| 2 | #include <vector> |
| 3 | using namespace std; |
| 4 | int find\_missing(vector<int> &nums) { |
| 5 | int left = 0, right = nums.size() - 1; |
| 6 | while (left < right) { |
| 7 | int mid = left + (right - left) / 2; |
| 8 | if (nums[mid] - mid + ①) { |
| 9 | ②; |
| 10 | }else{ |
| 11 | ③; |
| 12 | } |
| 13 | } |
| 14 | return ➃; |
| 15 | } |
| 16 |  |
| 17 | int main() { |
| 18 | int n; |
| 19 | cin >> n; |
| 20 | vector<int> nums(n); |
| 21 | for (int i = 0; i < n; i++) |
| 22 | cin >> nums[i]; |
| 23 | int missing\_number = find\_missing(nums); |
| 24 | if (missing\_number == ⑤) { |
| 25 | cout << "Sequence is consecutive" << |
| 26 | endl; |
| 27 | }else{ |
| 28 | cout << "Missing number is " << missing\_number | << endl; |
| 29 | } |  |
| 30 | return 0; |  |
| 31 | } |  |

1. ①处应填（ ）

A. 1 B.nums[0] C.right D.left

1. ②处应填（ ）

A. left=mid+1 B.right=mid-1 C.right=mid D.left=mid

1. ③处应填（ ）

A.left=mid+1 B.right=mid-1 C.right=mid D.left=mid

1. ➃处应填（ ）

A.left+nums[0] B.right+nums[0] C.mid+nums[0] D.right+1

1. ⑤处应填（ ）

A.nums[0]+n B.nums[0]+n-1 C.nums[0]+n+1 D.nums[n-1]

1. （编辑距离）给定两个字符串，每次操作可以选择删除（Delete）、插⼊（Insert）、替换（Replace），⼀个字符，求将第⼀个字符串转换为第⼆个字符串所需要的最少操作次数。
   1. #include <iostream>
   2. #include <string>
   3. #include <vector>
   4. using namespace std;
   5. int min(int x, int y, int z) {
   6. return min(min(x, y), z); 7 }

8

9 int edit\_dist\_dp(string str1, string str2) {

1. int m = str1.length();
2. int n = str2.length();
3. vector<vector<int>> dp(m + 1, vector<int>(n + 1)); 13 for (int i = 0; i <= m; i++) {

14 for (int j = 0; j <= n; j++) { 15 if (i == 0)

16 dp[i][j] = (1);

17 else if (j == 0)

18 dp[i][j] = (2);

19 else if ((3))

20 dp[i][j] = (4);

21 else

22 dp[i][j] = 1 + min(dp[i][j - 1], dp[i - 1][j], (5));

23 }

24 }

25 return dp[m][n]; 26 }

27

1. int main() {
2. string str1, str2;
3. cin >> str1 >> str2;
4. cout << "Mininum number of operation:"
5. << edit\_dist\_dp(str1, str2) << endl;
6. return 0; 34 }
7. ①处应填（ ）

A.j B.i C.m D.n

1. ②处应填（ ）

A.j B.i C.m D.n

1. ③处应填（ ）

A. str1[i-1]==str2[j-1] B. str1[i]==str2[j] C. str1[i-1]!=str2[j-1] D. str1[i]!=str2[j]

1. ➃处应填（ ）

A. dp[i-1][j-1]+1 B. dp[i-1][j-1]

C. dp[i-1][j] D. dp[i][j-1]

1. ⑤处应填（ ）

A. dp[i][j] + 1 B. dp[i-1][j-1]+1

C. dp[i-1][j-1] D. dp[i][j]