2023年CCF非专业级别软件能力认证第一轮 （CSP-J）入门级C++语言试题

题目总数：20　总分数：100

一、单项选择题

第 1 题　单选题

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？（ ）。

A. unsigned

B. const

C. static

D. mutable

答案　B

解析　const对象一旦被创建就不能修改其值，所以const对象必须进行初始化。

第 2 题　单选题

八进制数$12345670_{(8)}$和$07654321_{(8)}$的和为（ ）。

A. $22222221_{(8)}$

B. $21111111_{(8)}$

C. $22111111_{(8)}$

D. $22222211_{(8)}$

答案　D

解析　直接按照八进制进行计算，即逢八进一。

$$\begin{array}{r} (12345670)_8 \\ +\ (07654321)_8 \\ \hline (22222211)_8 \end{array}$$

第 3 题　单选题

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（ ）。

```
union Data{
    int num;
```

```
    float value;
    char symbol;
};
union Data data;
```

A. datvalue = 3.14;

B. value.data = 3.14;

C. data->value = 3.14;

D. value->data = 3.14;

答案 A

解析 union即为联合，它是一种特殊的类。访问其成员变量和结构体类似，使用成员运算符.进行访问，即：联合类型变量名.成员变量名。

第 4 题 单选题

假设有一个链表的节点定义如下：

```
struct Node {    int data;    Node* next;};
```

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？（）

A. Node* newNode = new Node; newNode->data = 42; newNode->next = head; head = newNode;

B. Node* newNode = new Node; head->data = 42; newNode->next = head; head = newNode;

C. Node* newNode = new Node; newNode->data = 42; head->next = newNode;

D. Node* newNode = new Node; newNode->data = 42; newNode->next = head;

答案 A

解析 因为newNode要成为第一个结点，所以newNode指向下一个的结点就是头结点，然后把newNode赋值给head，保证这一点即可。

第 5 题 单选题

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（）。

A. 6

B. 7

C. 8

D. 9

答案 C

解析 问高度至少是多少，所以每层都要尽可能满，所以找第一个 $3^0+3^1+3^2+...+3^i>=2023$ 的i，即为答案。

第 6 题　单选题

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。

A. 31

B. 18

C. 21

D. 33

答案　B

第 7 题　单选题

以下关于高精度运算的说法错误的是（ ）。

A. 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。

B. 大整数除以小整数的处理的步骤可以是，将被除数和除数对齐，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。

C. 高精度乘法的运算时间只与参与运算的两个整数中长度较长者的位数有关。

D. 高精度加法运算的关键在于逐位相加并处理进位。

答案　C

解析　高精*高精的运算时间与参与运算的两个大整数的长度乘积有关。

第 8 题　单选题

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

答案　A

解析　中缀表达式转后缀表达式的方法：
①将所有运算按照优先级加上小括号；
②将所有运算符移到对应小括号的后面；
③去掉小括号；
可以按照上述方法，从选项进行推导容易得到答案为A。

第 9 题　单选题

数$101010_{(2)}$和$166_{(8)}$的和为（ ）。

A. $10110000_{(2)}$

B. $236_{(8)}$

C. $158_{(10)}$

D. $A0_{(16)}$

答案 D

解析 (101010)2 + (166)8 = (42)10+ (118)10 = (160)10，(160)10=(10100000)2=(240)8=(A0)16。

第 10 题　单选题

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

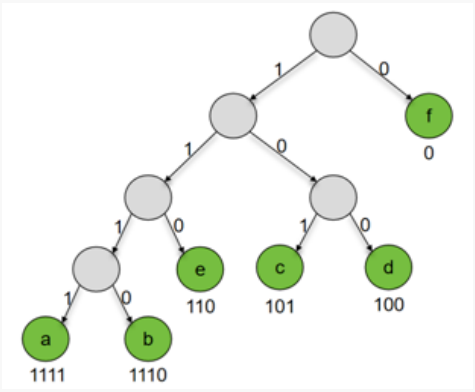
B. 1010，1001，1000，011，010，00

C. 000，001，010，011，10，11

D. 1010，1011，110，111，00，01

答案 A

解析 按照哈夫曼编码规则可以画出哈夫曼树，每个结点的哈夫曼编码如下图所示：



第 11 题　单选题

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？（）

A. EDBGFCA

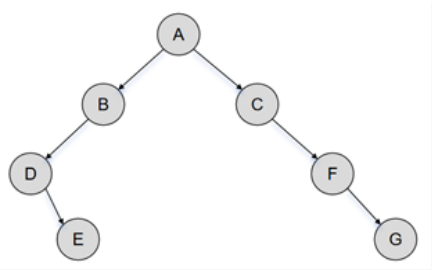
B. EDGBFCA

C. DEBGFCA

D. DBEGFCA

答案 A

解析 可以根据前序遍历和中序遍历画出对应的二叉树，如下图所示：



根据上图可知，对应的后序遍历是：EDBGFCA。

第 12 题　单选题

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

答案　B

解析　访问2,3之前要先访问1，访问4之前要先访问3或4，所以一个有效的拓扑序列为：1,2,3,4。

第 13 题　单选题

在计算机中，以下哪个选项描述的数据存储容量最小？（ ）

A. 字节（byte）

B. 比特（bit）

C. 字（word）

D. 千字节（kilobyte）

答案　B

解析　计算机最小的存储单位是：比特(bit)。
字节(Byte)是计算机用于计量存储容量的一种计量单位。字节中又分为：字节(Byte)、千字节(KB)、兆字节(MB)、吉字节(GB)、太字节(TB)、拍字节(PB)。
字是指在计算机中能够被处理和存储的最小单位。一般来说，一个字由若干个比特(bit)组成。

第 14 题　单选题

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

答案 A

解析 至少包括一个女生，最多包含3个女生，所以可以分为三种情况：
①包含一个女生，有C(12,1)·C(10,2) = 540种方法；
②包含两个女生，有C(12,2)·C(10,1) = 660种方法；
③包含三个女生，有C(12,3) = 220种方法；
因此，共有540 + 660 + 220 = 1420 种方案。

第 15 题 单选题

以下哪个不是操作系统？（ ）

A. Linux

B. Windows

C. Android

D. HTML

答案 D

解析 Linux、Windows和Android都是操作系统，而HTML属于超文本标记语言。

二、阅读程序题

第 16 - 20 题 组合题

```
#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：

第 16 题　判断题

当输入为“2 2 2”时，输出为“1.7321”（ ）

A. 正确

B. 错误

答案 A

第 17 题　判断题

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（ ）

A. 正确

B. 错误

答案 A

第 18 题　判断题

程序总是输出四位小数（ ）

A. 正确

B. 错误

答案 A

第 19 题　单选题

当输入为“3 4 5”时，输出为（ ）

A. "6.0000"

B. "12.0000"

C. "24.0000"

D. "30.0000"

答案 A

第 20 题　单选题

当输入为“5 12 13”时，输出为（ ）

A. "24.0000"

B. "30.0000"

C. "60.0000"

D. "120.0000"

答案 B

解析

本程序较为简洁，最重要的部分是函数 f。阅读 f 的返回值计算方式可知，这是计算三角形面积的海伦公式，其将 a、b、c 作为浮点数处理，考虑到了三角形面积可能非整数的情形。

值得注意：主函数中用到了平时学习较为少见的 cout 处理方式，其中 cout.flags(ios :: fixed) 表示以定点形式显示浮点数，如果没有这一句，则输出会形如 1e4 这样的科学计数法，而 cout.precision(4) 则是指定小数位数为 4 位。这里有一个坑：常用的 setprecision 方法，在输出浮点数时，不会保留尾随 0，而本程序中以上两句的结合，会对 0.1 这样的浮点数，输出 0.1000，其携带尾随 0。

最后，本题并没有明确 a、b、c 是否一定能够组成一个三角形。

（1）【解析】由以上分析，可知要输出边长为 2 的等边三角形的面积，其数学表达应为。如果能够记得常用根式的小数点后展开，则这一问可以快速解决，否则则需依次反向计算 1.7320、1.7321 和 1.7322 的平方，观察哪一项最接近 3。

（2）【解析】乘法满足交换律，且即使从程序的角度来考虑，double 类型的 s、a、b、c 的计算在输入不超过 1000 时，不会发生溢出。所以 sqrt 开根号的值不变，不会改变程序的运行结果。

（3）【解析】由以上分析，即使结果可以省去末尾的 0，也依然会输出为 4 位小数。实际上这一特性可以由 19、20 两问看出。

（4）【解析】边长为 3、4、5 的组合是非常典型的直角三角形，3 和 4 是其直角边，因此面积为，在程序中还要再加上小数点后 4 位0。

（5）【解析】与 19 问类似，5、12、13 也是非常经典的直角三角形三边组合，因此面积为，再加上小数点后 4 位0。

第 17 - 22 题　组合题

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

int f(string x,string y){
    int m=x.size();
    int n=y.size();
    vector<vector<int>>v(m+1,vector<int>(n+1,0));
   for(int i=1;i<=m;i++){
       for(int j=1;j<=n;j++){
            if(x[i-1]==y[j-1]){
                v[i][j]=v[i-1][j-1]+1;
            }else{
                v[i][j]=max(v[i-1][j],v[i][j-1]);
            }
        }
    }
    return v[m][n];
}

bool g(string x,string y){
    if(x.size() != y.size()){
        return false;
    }
    return f(x+x,y)==y.size();
```

```
27  }
28
29  int main(){
30      string x,y;
31      cin>>x>>y;
32      cout<<g(x,y)<<endl;
33      return 0;
34  }
```

第 17 题　判断题

f函数的返回值小于等于min（n,m）。

A. 正确

B. 错误

答案　A

第 18 题　判断题

f函数的返回值等于两个输入字符串的最长公共子串的长度。（ ）

A. 正确

B. 错误

答案　B

第 19 题　判断题

当输入两个完全相同的字符串时，g函数的返回值总是true（ ）

A. 正确

B. 错误

答案　A

第 20 题　单选题

将第19行中的“v[m][n]”替换为“v[n][m]”，那么该程序（ ）

A. 行为不变

B. 只会改变输出

C. 一定非正常退出

D. 可能非正常退出

答案　D

第 21 题　单选题

当输入为“csp-j p-jcs”时，输出为：（ ）

A. "0"

B. "1"

C. "T"

D. "F"

答案 B

第 22 题 单选题

当输入为“csppsc spsccp”时，输出为：（ ）

A. "T"

B. "F"

C. "0"

D. "1"

答案 D

解析

程序以 f 和 g 两个函数为主，其中对 STL 的考察与去年类似——f 函数中使用到了 vector 套 vector 的用法，部分同学可能学习过 vector 容器的用法，但二维 vector 可能一时无法理解。然而，对 STL 的“冷门”考察，一定能够从程序的上下文中找到答案。只要观察接下来 v 的用法，就会发现，可以把它当做是一个二维数组。

f 函数实际考察动态规划中的最长公共子序列（LCS），作为一个经典知识点，这里省去推导，直接给出结论：v[i][j] 表示字符串 x 的第 1~i 位和字符串 y 的第 1~j 位的最长公共子序列的长度。对于一个字符串，其子序列指的是可以在字符串中任意删掉一些字符后剩下的部分，比如 bd 是abcd 的子序列，c 和 a 也是abcba 的子序列，公共子序列则要求同时是两个字符串的子序列。

最后，g 函数先筛去两个字符串长度不一致的情况，然后对于字符串 x，在其末尾再复制一份，这一变换形如 abcd→abcdabcd，判断变换后的双倍 x 和 y 的最长公共子序列长度是否等于 y 的原长度。

（1）【解析】f 函数返回两个字符串的最长公共子序列长度，最好情况下，这一公共子序列也只能恰好等于其中某一字符串，其长度不能再超过字符串 x 和 y 的长度，而 m 和 n 分别代表 x 和 y 的长度，故正确。

（2）【解析】子串与子序列是两个不同的概念，其中子串要求字符必须在原字符串中连续出现，而子序列则只对顺序有要求。例如：ace 是 abcde 的子序列，但不是其子串；cab 既是abcabc 的子串，又是其子序列。实际上，一个字符串的子串，一定是原串的子序列，但反之不然。

（3）【解析】输入的 x 和 y 相同时，x+x 等于 y+y，其和 y 的最长公共子串恰为 y 本身，故 g 选项返回的判断条件成立。

（4）【解析】STL 容器有非常严格的使用要求，以一维情形举例，如果定义了 vector<int> a(100, 0)，表示 a 的下标范围是 0~99，此时如果下标访问了a[100]，则一定会使程序非正常退出，二维的情形类似。

如果替换为 v[n][m]，在 n = m，也就是两个字符串长度一致时，程序行为不变；但如果 n ≠ m，则会导致某一维的下标访问越界，此时一定非正常退出。综合而言，互换会导致可能非正常退出。

（5）【解析】此时将求 csp-jcsp-j 和 p-jcs 的最长公共子序列，可知后者恰为前者的子串，故 g 函数返回的判断条件成立。

注意输出的恰为 g 函数返回的 bool 值，需要排除 T 和 F。

（6）【解析】此时将求csppsccsppsc和spsccp 的最长公共子序列，从前者依次选出第 2、3、5、6、7、9 个字符，恰为 spsccp，可知后者是前者的子序列，故 g 函数返回的判断条件成立。

第 18 - 23 题 组合题

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

int solve1(int n){
    return n*n;
}

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;
}

int main(){
    int n;
    cin>>n;
    cout<<solve2(solve1(n))<<" "<<solve1((solve2(n)))<<endl;
    return 0;
}
```

第 18 题　判断题

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（ ）

A.　正确

B.　错误

答案　A

第 19 题　判断题

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（ ）

A.　正确

B.　错误

答案　A

第 20 题　判断题

如果输入的n为质数，solve2（n）的返回值为n2+1（ ）

A.　正确

B. 错误

答案 A

第 21 题 单选题

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

答案 B

第 22 题 单选题

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

答案 D

第 23 题 单选题

当输入为“5”时，输出为（ ）

A. "651.625"

B. "650.729"

C. "651.676"

D. "652.625"

答案 C

解析

本题考察约数。函数 solve1 返回参数 n 的平方；函数solve2 返回参数 n 的约数的平方和。主函数对两个函数复合调用，先输出 n 的平方的约数平方和，再输出 n 的约数平方和的平方。

（1）solve2 的作用是返回 n的所有约数的平方和。

（2）枚举约数时，如果i等于n/i，这两个表达式就代表同一个约数（即n的平方根），所以需要13行的判断避免它被重复计算。

（3）质数的约数只有1和自身，它们的平方和即为n2+1

（4）n=p2时，它的约数有1、p、p2，它们的平方和等于1+p2+p4，再利用p2=n 可化为n2+n+1。

（5）

对 n 分解质因数 $n = p_1^{\alpha 1} p_2^{\alpha 2} \cdots p_k^{\alpha k}$ 的情况。第一项 $= (1 + p_1^2 + p_1^4 + \cdots + p_1^{4\alpha_1}) \times \cdots \times (1 + p_k^2 + p_k^4 + \cdots + p_k^{4\alpha_k})$；第二项 $= (1 + p_1^2 + \cdots + p_1^{2\alpha_1})^2 \times \cdots \times (1 + p_k^2 + \cdots + p_k^{2\alpha_k})^2$ 。显然第一项小于第二项。最后考虑到 $n = 1$ 时两者相等，结论是小于等于0。

（6）n=5，第一项=1+25+625=651、第二项=(1+25) 2 =676。

三、完善程序

第 19 - 23 题　组合题

(寻找被移除的元素)问题:原有长度为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，导致长度为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。

```
#include <iostream
#include <vector>

using namespace std;

int find missing(vector<int>& nums) (
int left = 0, right = nums.size() - 1;
while (left < right){
  int mid = left + (right  left) / 2;
  if (nums[mid] - mid+      ①      ) (
            ②      ;
   }else{
           ③
   }
  }
 return       ④      ;
}

int main() (
   int n;
cin >> n;
vector<int> nums(n);
for (int i= 0; i< n; i++) cin >> nums[i];
int missing_number = find_missing(nums);
if_(missing_number ==       ⑤      ) {
    cout << "Sequence is consecutive" << endl;
}else{
   cout << "Missing number is " << ,missing numbeer << endl;
}
return 0;
}
```

第 19 题　单选题

①处应填（ ）

A.　1

B.　nums[0]

C.　right

D.　left

答案　B

第 20 题　单选题

②处应填（ ）

A.　left=mid+1

B.　right=mid-1

C.　right=mid

D.　left=mid

答案　A

第 21 题　单选题

③处应填（ ）

A.　left=mid+1

B.　right=mid-1

C.　right=mid

D.　left=mid

答案　C

第 22 题　单选题

④处应填（ ）

A.　left+nums[0]

B.　right+nums[0]

C.　mid+nums[0]

D.　right+1

答案　A

第 23 题　单选题

⑤处应填（ ）

A.　nums[0]+n

B.　nums[0]+n-1

C.　nums[0]+n+1

D. nums[n-1]

答案 D

解析

本题考察二分法。在本程序中find_missing函数就是利用二分法来找到一个长度为n的数组中不连续的位置，从而找出被移除元素的值。只需通过二分找到从左往右第一处使得nums[i]不为nums[0]+i的的位置，那么在数组中被移除的数就是nums[0]+i。

（1）若数组连续，一定有nums[i]==nums[0]+i，所以只需通过二分找到第一处使得nums[i]不为nums[0]+i的的位置即可。因此二分的判断条件是nums[mid]==nums[0]+mid。选B

（2）若满足33所填的判断条件，则说明此时的mid猜小了，实际不连续位置（第一处使得nums[i]不等于nums[0]+i的位置）一定在mid之后，所以需要更新二分的左端点，并更新为mid+1，因此选择A

（3）若不满足33所填的判断条件，就说明mid有可能猜大了，实际不连续位置（第一处使得nums[i]不等于nums[0]+i的位置）一定不在mid之后，可以等于mid。因此需要更新二分右端点，并且更新为mid。答案为C

（4）左端点left才能表示最终二分找到的答案（也即不连续位置），因此最终被移除的数就是left+nums[0]。首先很容易排除D，然后因为mid只出现在二分的过程中，而且mid的定义也在while里，也很容易排除C。

可能的问题：这里为何选择A而不选择B，事实上我们二分模板中的书写习惯是return left。在一些情况下return right会出现问题，但在本题中实际上最后一定是left==right，所以理论上填B也可以。

（5）当n个数字连续时，最终我们会得到，left=n-1，right=n-1。由36所填代码，我们返回的数字就是nums[0]+n-1与不连续位置出现在第n个数字返回的数值相同。因此我们需要对这两种情况进行讨论，如果n个数字连续，会满足nums[n-1]==nums[0]+n-1，如果是刚好不连续位置在第n个数的话不会使得这个式子成立。因此如果返回数字等于nums[n-1]的话，就说明原来n个数连续。故而选择D。

第 20 - 24 题 组合题

（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。

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

int min(int x,int y,int z){
return min(min(x,y),z);
}

int edit_dist_dp(string str1,string str2){
int m=str1.length();
int n=str2.length();
vector<vector<int>> dp(m+1,vector<int>(n+1));

for(int i=0;i<=m;i++){
for(int j=0;j<=n;j++){
if(i==0)
dp[i][j]=     ①     ;
else if(j==0)
```

```
dp[i][j]=     ②     ;
else if(     ③     )
dp[i][j]=     ④     ;
 else
 dp[i][j]=1+min(dp[i][j-1],dp[i-1][j],     ⑤     );
 }
 }
return dp[m][n];
}

int main(){
 string str1,str2;
 cin>>str1>>str2;
 cout<<"Mininum number of operation:"
 <<edit_dist_dp(str1,str2)<<endl;
 return 0;
}
```

第 20 题　单选题

①处应填（ ）

A. j

B. i

C. m

D. n

答案　A

第 21 题　单选题

②处应填（ ）

A. j

B. i

C. m

D. n

答案　B

第 22 题　单选题

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]

答案　A

第 23 题　单选题

④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]

答案　B

第 24 题　单选题

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]

答案　C

解析

先阅读 main 函数，可以得知代码读入了两个字符串之后输出了 edit_dist_dp 的返回值，因此重点关注 edit_dist_dp 函数；

11-13 行定义了 n 为 str1 的长度，m 为 str2 的长度，一个二维的 vector（可以当成二维数组）dp[0..m][0..n]；

15-26 行为 dp 过程；

27 行为返回 dp[n][m]，看完这一行就应该明白 dp 数组的含义，之后通过 dp 状态将 15-26 行的空填出来；dp[n][m] 表示将 str1 长度为 n 的前缀变成 str2 长度为 m 的前缀需要的最少步数，特别注意 str1, str2 是 string，下标范围分别是 [0..m-1] 与 [0..n-1]。

（1）考察 dp[0][j] 的值，根据阅读得到的 dp 数组的含义，dp[0][j] 表示 str1 长度为 0 的前缀（空字符串）变成 str2 长度为 j 的前缀的最少步数，这里明显最少在空串内添加 str2 的前 j 个字符，因此填 j选 A 选项

（2）考察 dp[i][0] 的值，根据阅读得到的 dp 数组的含义，dp[i][0] 表示 str1 长度为 i 的前缀变成str2 长度为 0 的前缀（空字符串）的最少步数，这里明显最少是将 str1 的前 i个字符全部删除，因此填 i 选 B 选项

（3）考察编辑距离 dp 转移，需要阅读 22-24 行的代码可知这里应该是两个字符相等不需要操作的情况，也就是 str1 的第 i个字符与 str2 的第 j 个字符相等的情况，但是要特别注意的是这一问埋了大坑，str1 的第 i个字符是str1[i-1]，不要跳进去了，选 A

（4）40题时已经说过，如果两个字符相等的话，不需要操作，因此将 str1 前 i个字符变成 str2 前 j 个字符需要的最少步数就与将 str1 前 i-1 个字符变成 str2 前j-1 个字符是一样的，填 dp[i-1][j-1]，选 B

（5）这里有一个在上面定义的 min 函数，功能是对三个整数求最小值，观察第 24 行 dp[i][j] = 1 + min(dp[i][j - 1], dp[i - 1][j], ?) 由前面的 1 + 可知这里进行了一次操作，那么dp[i][j - 1] 就对应着插入，dp[i - 1][j] 对应着删除，剩下要填的就是替换了，填 dp[i-1][j-1]，选 C