CSP 2023 入门级第一轮答案与解析

一、单项选择（一题2分，共15题，共计30分）

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?

A. unsigned B. const C. static D. mutable

答案：B

解析：在 C++ 中，关键字 const 用于声明一个常量，即其值不能被修改。一旦将变量声明为 const ，它的值将在声明后保持不变，不能再被修改。这是一种保护数据完整性和避免意外修改的方式。可以使用 const 来声明任何类型的变量，包括基本类型、类对象、指针以及引用等。

其他选项：

unsigned 是用于修饰整数类型，表示该整数类型只能表示非负数。

static 用于静态变量和静态成员函数的声明，它们的特点是在程序执行期间只有一个实例。

mutable 用于成员变量的声明，表示该变量可以在 const 成员函数中被修改。

2.八进制数(12345670)8和(07654321)8 的和为？

A.222222218 B.211111118 C.221111118 D.222222118

答案：D

解析：八进制加减法的方法与十进制加减法类似，但需要注意进位与退位的问题。下面通过例子来说明：八进制加法

例如，将八进制数“165”和“25”相加：

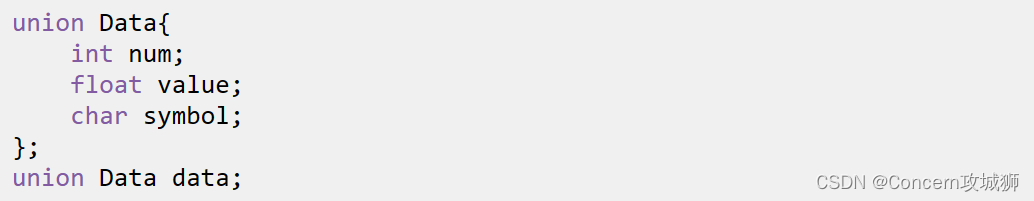
1+2 = 3（个位数）

6+5 = 13（十位数，需要进位）

1+2+1= 4（百位数，需要进位）

因此，八进制数“165”和“25”相加的结果为“232”。

3.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？



A. data.value = 3.14;

B. value.data = 3.14;

C. data -> value = 3.14;

D. value->data = 3.14;

答案：A

解析： data.value = 3.14这样可以直接将 3.14 赋值给 data 的 value 成员。请确保 data 的类型具有名为 value 的公共成员变量，且其数据类型与 3.14 匹配（例如 float 或 double）。

1. 假设有一个链表的节点定义如下：

struct node{

int data;

Node\* next;

};

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：D选项没有更新 head 的值，导致没法插入新的数据，BC 选项对 head 的 data 进行修改

5.根节点的高度为 1，一棵拥有 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：对于一棵拥有2023个节点的三叉树，我们可以计算其最小高度。

三叉树是一种每个节点最多有三个子节点的树结构。在这棵三叉树中，根节点的高度为1，意味着从根节点到任意叶子节点的最短路径长度为1。

现在我们需要确定这棵三叉树的最小高度。假设它的最小高度为h。

对于一棵树而言，节点数n与最小高度h之间存在以下关系： n = 1 + 3 + 3^2 + 3^3 + ... + 3^(h-1) （等比数列求和）

将该等比数列求和公式代入，并解得h： 2023 = (3^h - 1) / 2

有多种方法可以求解这个方程，例如通过逐渐增加h的值进行尝试，直到等式左侧大于2023，即可确定最小的合法h值为所求。

6.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

解析：首先，我们可以列举出满足条件的情况：选择一个时间段练习，其余时间段都是空闲时间段。这种情况共有7种。选择两个相邻的时间段练习，其余时间段都是空闲时间段。这种情况共有6种。选择三个相邻的时间段练习，其余时间段都是空闲时间段。这种情况共有5种。

7.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：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。

实际上，高精度乘法的运算时间与两个整数的位数之积相关。具体来说，假设第一个整数有n位，第二个整数有m位，则高精度乘法的运算时间为O(nm)。因此，与参与运算的两个整数中长度较长者的位数有关，而不仅仅与长度较长者的位数有关。

其他选项中的说法是正确的： 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 D. 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

答案：A

解析：我们可以从后缀表达式的左到右扫描每个数字和操作符。遇到数字时，我们将其压入栈中；遇到操作符时，我们从栈顶弹出两个数字，进行相应的计算，并将结果压回栈中。重复此过程直到处理完所有的操作符和数字，最后栈中剩下的就是中缀表达式的结果。

具体来说，我们可以模拟一下这个过程：

遇到数字 6，将其压入栈中。 栈：[6]

遇到数字 2，将其压入栈中。 栈：[6, 2]

因此，最终的中缀表达式是 ((6-(2+3))\*(3+(8/2)))^2+3。

遇到数字 3，将其压入栈中。 栈：[6, 2, 3]

遇到操作符 +，从栈顶弹出两个数字 2 和 3，计算 2+3=5，将结果 5 压入栈中。栈：[6, 5]

遇到操作符 -，从栈顶弹出两个数字 5 和 6，计算 6-(5)=1，将结果 1 压入栈中。 栈：[1]

遇到数字 3，将其压入栈中。 栈：[1, 3]

遇到数字 8，将其压入栈中。 栈：[1, 3, 8]

遇到数字 2，将其压入栈中。 栈：[1, 3, 8, 2]

遇到操作符 /，从栈顶弹出两个数字 2 和 8，计算 8/2=4，将结果 4 压入栈中。 栈：[1, 3, 4]

遇到操作符 +，从栈顶弹出两个数字 4 和 3，计算 3+4=7，将结果 7 压入栈中。 栈：[1, 7]

遇到操作符 ，从栈顶弹出两个数字 7 和 1，计算 17=7，将结果 7 压入栈中。 栈：[7]

遇到数字 2，将其压入栈中。 栈：[7, 2]

遇到操作符 ^，从栈顶弹出两个数字 2 和 7，计算 7^2=49，将结果 49 压入栈中。 栈：[49]

遇到数字 3，将其压入栈中。 栈：[49, 3]

遇到操作符 +，从栈顶弹出两个数字 3 和 49，计算 49+3=52，得到最终结果。 栈：[52]

因此，最终的中缀表达式是 ((6-(2+3))\*(3+(8/2))^2+3。

9.数1010102和1668的和为？

A.(10110000)2

B.(236)8

C.(158)10

D.(A0)16

答案：D

解析：没啥好说的，小学生都会

10.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abcdef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：根据字符 {a, b, c, d, e, f} 对应的频率分别为 5%, 9%, 12%, 13%, 16%, 45%，我们可以构建哈夫曼树来找到正确的哈夫曼编码。

首先，我们按照频率从小到大排序字符：

a: 5%

b: 9%

c: 12%

d: 13%

e: 16%

f: 45%

然后，我们开始构建哈夫曼树：

频率最小的字符 a 和 b 构成一组，合并后的频率为 5% + 9% = 14%。

合并后的节点与下一个频率最小的字符 c 组成一组，合并后的频率为 14% + 12% = 26%。

合并后的节点与下一个频率最小的字符 d 组成一组，合并后的频率为 26% + 13% = 39%。

合并后的节点与下一个频率最小的字符 e 组成一组，合并后的频率为 39% + 16% = 55%。

最后，合并剩余的两个字符 f 和合并后的节点，合并后的频率为 55% + 45% = 100%。

接下来，我们可以根据哈夫曼树的结构得到对应的编码：

a: 1111 b: 1110 c: 101 d: 100 e: 110 f: 0

因此，选项 A 中的编码 {1111, 1110, 101, 100, 110, 0} 是字符 abcdef 对应的一组哈夫曼编码。

11.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：根据给定的前序遍历 "ABDECFG" 和中序遍历 "DEBACFG"，我们可以重建二叉树。首先，根据前序遍历的第一个元素 "A"，确定根节点为 "A"。然后，在中序遍历中找到根节点 "A"，并以它为界，将中序遍历分为左子树和右子树。

根据中序遍历结果 "DEBACFG"，我们可以得到以下划分： 左子树的中序遍历结果：DEB 右子树的中序遍历结果：CFG

接下来，我们分析前序遍历的结果 "ABDECFG"： 从前序遍历的第二个字符开始 "B"，可以确定它是根节点的左子节点。 然后，在左子树的中序遍历结果 "DEB" 中，我们可以确定根节点的左子节点是 "D"，右子节点是 "E"。

继续分析前序遍历的结果 "ABDECFG"： 在剩余的字符中 "CFG"，可以确定它们是根节点的右子节点。 在右子树的中序遍历结果 "CFG" 中，我们可以确定根节点的左子节点是 "C"，右子节点是 "F"。

最终，我们得到了以下重建的二叉树的结构：



根据二叉树的后序遍历特点，我们可以得到后序遍历结果为 "ADEBFGC"。因此，选项 A 中的后序遍历结果 EDBGFCA" 是这棵树的正确后序遍历结果。

12.考虑一个有向无环图，该图包含 44 条有向边：(1,2),(1,3),(2,4)(1,2)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：给定有向无环图的边集为{(1,2), (1,3), (2,4), (3,4)}。有效的拓扑排序是指按照拓扑顺序依次排列图中的节点，使得对于每条有向边 (u,v)，节点 u 在拓扑序列中出现在节点 v 之前。

根据给定的边集，我们可以得到节点之间的关系：

1 -> 2 -> 4

-> 3 -> 4

我们可以观察到节点 1 是没有入度的，因此在拓扑排序中应该是第一个节点。节点 2 和节点 3 都有一个入度，因此它们应该位于节点 1 之后。节点 4 是拓扑顺序中的最后一个节点，因为它没有出度。

选项 B 中的拓扑排序 "1,2,3,4" 是这个有向无环图的一个有效拓扑排序。

13.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：比特（bit）描述的是计算机中最小的数据存储单位，它代表了二进制位，即0或1。字节（byte）是计算机中常用的数据存储单位，它由8个比特组成，可以保存一个字符或者一个字母。字（word）是在不同的上下文中使用的术语，它通常表示机器字的大小，字的大小因计算机体系结构而异，可以是8位、16位、32位或者64位。千字节（kilobyte）是数据存储容量的一种更大的单位，等于1024个字节。

因此，从描述的数据存储容量最小来看，选项 B 中的比特（bit）所表示的数据存储容量最小。

14.一个班级有 1010 个男生和 1212 个女生。如果要选出一个 33 人的小组，并且小组中必须至少包含 11 个女生，那么有多少种可能的组合?()

 A. 14201420

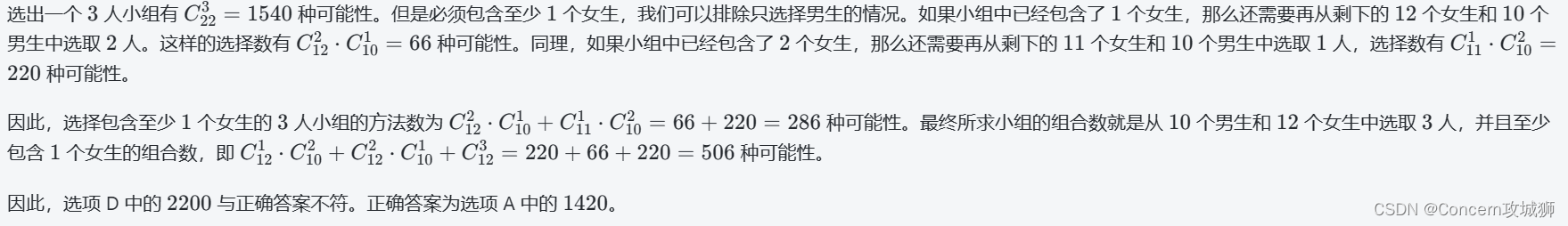
 B. 17701770

 C. 15401540

 D. 22002200

答案：A

解析：分式不好写，直接上图片



15.以下哪个不是操作系统?()

A. Linux

B. Windows

C. Android

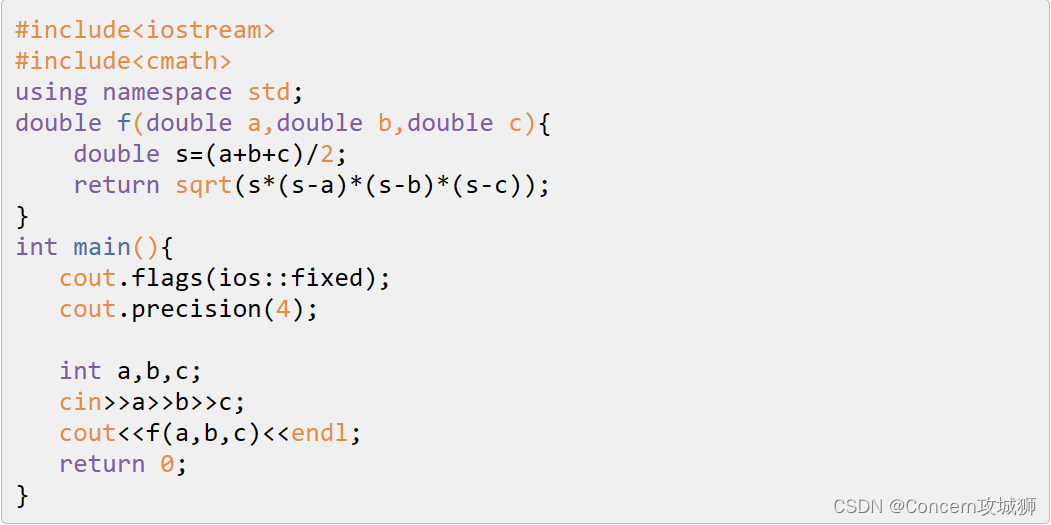
D. HTML

答案：D

解析：送命题。。。不说了。。。谁答不对就抄1000遍。

二、阅读程序（程序输入不超过数组或字符串定义的范围；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）

16.①、



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

1、（2分）当输入为 2 2 2 时，输出为1.7321（ ）

2、（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（ ）

3、（2分）程序总是输出四位小数（ ）

单选题

4、当输入为 3 4 5 时，输出为（ ）

5、当输入为 5 12 13 时，输出为（ ）

选项

1. A. 正确 B. 错误

2. A. 正确 B. 错误

3. A. 正确 B. 错误

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

5. A. 24.0000 B. 30.0000 C. 60.0000 D. 120.0000

答案：A A B A B

解析：

1、当输入为 2 2 2 时，根据海伦公式计算三角形的面积，结果是 1.7321（四舍五入到小数点后四位）。根据您提供的代码，程序的输出将确实是 1.7321。

2、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。这是因为乘法操作符具有交换律，对于任意实数 a 和 b，a \* b 的结果与 b \* a 的结果相同。

在这个特定的计算中，(s-b)\*(s-c) 和 (s-c)\*(s-b) 的结果是相同的，因为它们只是交换了两个乘法操作数的顺序。因此，无论您选择哪个顺序，都不会对最终的计算结果产生影响。

3、这个程序并不总是输入四位小数。在主函数的第13行，使用 cout.precision(4) 设置了 cout 流的小数点精度为 4，这意味着程序将以四位小数输出结果。然而，如果计算结果本身具有更少的小数位数，那么程序将按照实际的小数位数进行输出。例如，如果计算结果是 3.14159，那么程序会输出五位小数。因此，程序的输出取决于实际的计算结果。如果计算结果具有不超过四位小数的精度，程序将输出相应的小数位数。只有当计算结果具有超过四位小数位时，设置 cout.precision(4) 才会生效，将结果截断为四位小数。

4、当输入为 3 4 5 时，根据海伦公式计算三角形的面积，结果是 6.0000。根据给出的代码，程序的输出将是 6.0000。

由于在主函数的第13行设置了 cout.precision(4)，输出结果会保留四位小数。因此，程序将输出 6.0000。

5、代码中定义了一个函数 f，它接收三个参数 a、b、c，表示三角形的三条边长。在 f 函数内部，使用海伦公式计算三角形的面积，并将结果返回。

在主函数中，使用 cout.flags(ios::fixed) 设置输出流的格式为固定小数点格式。然后使用 cout.precision(4) 设置小数点精度为四位小数。

接下来，通过 cin 语句获取用户输入的三个整数 a、b、c，分别表示三角形的三条边长。然后调用函数 f(a, b, c) 计算三角形的面积，并使用 cout 输出结果。

因此，当输入为 5 12 13 时，根据海伦公式计算得到的三角形的面积为 30.0000，程序将输出 30.0000。

1. ②、



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长公共子串的长度。（）

3、当输入两个完全相同的字符串时，g函数的返回值总是turn.( )

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csp-j  p-jcs时，输出为（）。

6、当输入为csppsc spsccp时，输出为（）。

选项

1. A. 正确 B. 错误

2. A. 正确 B. 错误

3. A. 正确 B. 错误

4. 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

5. A.0 B. 1 C. T D. F

6. A.T B. F C. 0 D. 1

答案：A B A D B D

解析：

1、f函数返回的是字符串x和y的最长公共子序列的长度，动态规划算法中使用了一个二维数组v[m+1][n+1]来记录最长公共子序列的长度，m和n分别表示字符串x和y的长度。在每次迭代中，只需要保留v数组的两行（即当前行和上一行）来计算下一行，因此空间复杂度为O(min{n, m})。由于最长公共子序列的长度不会超过字符串x和y的长度，所以f函数的返回值小于等于min{n, m}。

2、f函数的返回值表示的是最长公共子序列（Longest Common Subsequence）的长度，而不是最长公共子串（Longest Common Substring）的长度。最长公共子序列是指两个字符串中以相同顺序出现的字符组成的最长子序列，这些字符在原字符串中不一定连续。而最长公共子串则是指两个字符串中连续出现的相同字符组成的最长子串。

因此，f函数的返回值并不一定等于两个输入字符串的最长公共子串的长度。

3、因为在g函数的实现中，会先判断输入的两个字符串的长度是否相等。如果长度相等，接下来会调用函数f来计算最长公共子序列的长度。当两个字符串完全相同时，它们的最长公共子序列就是它们自身，长度等于字符串本身的长度。所以最后比较得到的最长公共子序列的长度与字符串y的长度相等，返回值为true。

4、如果将第19行中的v[m][n]替换为v[n][m]，那么程序将会出现错误。

在动态规划算法中，使用二维数组v来记录最长公共子序列的长度。数组v的维度是(m+1) x (n+1)，其中m和n分别表示两个字符串的长度。在算法的实现过程中，需要根据字符串的索引使用正确的数组下标进行访问。因为m和n表示字符串的长度，所以正确的访问方式是v[m][n]，而不是v[n][m]。

如果将第19行中的v[m][n]替换为v[n][m]，将导致访问数组越界，程序将出现错误。所以选择D.可能非正常退出

5、在 main 函数中，程序首先读取了两个字符串 x 和 y。然后调用函数 g(x, y) 来判断它们是否满足循环相似的条件。在函数 g 中，它将字符串 x 进行了拼接（x+x），然后调用函数 f(x+x, y) 来计算最长公共子序列的长度。如果最长公共子序列的长度等于字符串 y 的长度，则返回 true，否则返回 false。

对于输入 "csp-j p-jcs"，经过计算，在调用 f(x+x, y) 时，得到的最长公共子序列长度为 5，而字符串 y 的长度也为 5。因此，g(x, y) 返回 true。所以，输出应该是 1。

6、程序中的 g 函数用于判断两个字符串 x 和 y 是否符合循环相似的条件，即将字符串 x 做循环平移后是否等于字符串 y。在 main 函数中，分别读取了 x 和 y 的输入，然后调用 g(x, y) 来检查它们是否符合循环相似的条件。函数 g 中调用了函数 f 来计算最长公共子序列的长度，如果最长公共子序列的长度等于字符串 y 的长度，则返回 true，否则返回 false。

对于输入 "csppsc" 和 "spsccp"，它们符合循环相似的条件，因为字符串 "csppsc" 可以循环平移成为 "scsppc"，而 "scsppc" 与字符串 "spsccp" 是一样的。因此，g(x, y) 的结果为 true。在计算最长公共子序列的过程中，可以得到 "spsc" 是它们的最长公共子序列，它的长度为 4，而字符串 y 的长度也为 4。因此，f(x+x, y) 的结果为 4，g(x, y) 的结果为 true，所以输出应为 1。

1. ③、
2. 

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

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（ ）

5、当输入为正整数时，第一项减去第二项的差值一定（ ）

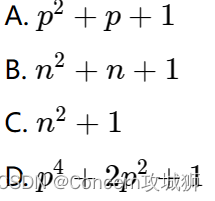
6、当输入为5时，输出为（ ）

选项

1. A. 正确 B. 错误

2. A. 正确 B. 错误

3. A. 正确 B. 错误

4.

5.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

6. A. 651.625

B. 650.729

C. 651.676

D. 652.625

答案：A A A B D C

三、完善程序（单选题，每小题 3 分，共计 30 分）

19.①(寻找被移除的元素)问题: 原有长度为 n+1 公差为 1 等差数列，将数列输到程序的数组时移除了一个元素，导致长度为 n 的连续数组可能不再连续，除非被移除的是第一个或最后一个元素。需要在数组不连续时，找出被移除的元素。试补全程序。

1、①处应填（ ）

2、②处应填（ ）

3、③处应填（ ）

4、④处应填（ ）

5、⑤处应填（ ）

选项

1、A. 1 B. nums[0] C. right D. left

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

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

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

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

答案：B A C A D

20.②（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;

}

1、①处应填（ ）

2、②处应填（ ）

3、③处应填（ ）

4、④处应填（ ）

5、⑤处应填（ ）

选项

1、A. j B. i C. m D. n

2、A. j B. i C. m D. n

3、

A. str1[i-1]==str2[j-1]

B. str1[i]==str2[j]

C. str1[i-1]!=str2[j-1]

D. str1[i]!=str2[j]

4、

A. str1[i-1]==str2[j-1]

B. str1[i]==str2[j]

C. str1[i-1]!=str2[j-1]

D. str1[i]!=str2[j]

5、

A. dp[i][j] + 1

B. dp[i-1][j-1]+1

C. dp[i-1][j-1]

D. dp[i][j]

答案：A B A B C