2023 年 GESP9 月认证 Python 六级试卷解析

CCF 编程能力等级认证，英文名 Grade Examination of Software Programming（以下简称 GESP），由中国计算机学会发起并主办，是为青少年计算机和编程学习者提供学业能力验证的平台。GESP 覆盖中小学全学段，符合条件的青少年均可参加认证。GESP 旨在提升青少年计算机和编程教育水平，推广和普及青少年计算机和编程教育。

GESP 考察语言为图形化（Scratch）编程、Python 编程及 C++编程，主要考察学生掌握相关编程知识和操作能力，熟悉编程各项基础知识和理论框架，通过设定不同等级的考试目标，让学生具备编程从简单的程序到复杂程序设计的编程能力，为后期专业化编程学习打下良好基础。

本次为大家带来的是 2023 年 9 月份 Python 六级认证真题解析。

一、单选题（每题 2 分，共 30 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

【答案】A

【解析】本题属于考察计算机基本知识。输入设备是一种用于向计算机或其他设备输入数据和信息的设备，它是用户和计算机系统之间进行信息交换的主要桥梁。常见的输入设备有键盘，鼠标，摄像头，扫描仪，光笔，手写输入板，游戏杆，语音输入装置等。故正确答案为 A 选项。

2.以下关于 Python 语言的描述，错误的是（ ）。

A. Python 提供了常用的数据结构，并支持面向对象编程

B. Python 是解释型语言

C. Python 是一种高级程序设计语言

D. Python 程序在运行前需要预先编译

【答案】D

【解析】本题属于考察面向对象的语言特点的知识。Python 是一种解释型语言，这意味着在运行 Python 程序时，不需要预先编译。相反，Python 解释器会逐行读取并执行代码。当遇到一个语法错误时，解释器会立即停止执行并报告错误。这种特性使得 Python 非常适合快速开发和调试，但也意味着它的执行速度相对较慢。故正确答案为 D 选项。

3.以下不属于面向对象程序设计语言的是（ ）。

A. C++

B. Python

C. Java

D. C

【答案】D

【解析】本题属于考察面向对象的思想。面向过程语言和面向对象程序设计语言是两种不同的编程范式，它们在处理问题和组织代码的方式上有着显著的区别。面向过程语言，如 C 语言，是一种基于程序和函数的编程范式。它强调的是解决问题的步骤和算法，通过将程序分解成一系列函数来完成任务。例如，下五子棋这个问题，面向过程的设计思路是首先分析解决这个问题的步骤：开始游戏、黑子先走、绘制画面、判断输赢等，然后用函数把这些步骤实现，在主函数里依次调用这些函数。面向对象程序设计语言，如 Python，是一种基于对象的编程范式。它将数据和处理数据的函数捆绑在一起形成一个对象，这个对象被称为类。在面向对象程序设计中，复杂的问题被拆解为一系列需要完成的任务，这些任务通过方法来实现。以五子棋为例，面向对象的设计思维可能会创建一个“Game”类来封装游戏的所有元素和行为。故正确答案为 D 选项。

4.下面有关 Python 类定义的说法，错误的是( )

A.Python 类实例化时，先执行 new()和 init()

B.Python 内置函数 bool()对于自定义类有效，必须在新定义类中定义 bool()函数

C.Python 自定义类不能适用于 for-in 循环

D.Python 自定义类可用 getitem()魔术方法定义方括号运算符

【答案】C

【解析】本题属于考察 Python 类的创建知识。Python 自定义类可以用于 for-in 循环，这是因为 Python 中的类是对象的抽象，而对象是类的实例。当我们创建一个自定义类的对象时，我们可以使用 for-in 循环来遍历该对象的属性和方法。故正确答案为 C 选项。

5.有关下面 Python 代码的说法，错误的是 （ ）。

```
class strReverse:
    def __init__(self, strData = ""):
        self.Data = strData
    def __repr__(self):
        return self.Data[::-1]

print(strReverse("ABC"))
```

A. 最后一行代码将输出 CBA

B. 最后一行代码将不能输出 CBA，因为没有定义 print()函数

C. 第 3 行代码的 Data 是 strReverse 类的数据属性

D. 最后一行代码将自动执行 init()函数

【答案】B

【解析】本题属于考察 Python 类的创建知识。这段代码定义了一个名为 strReverse 的类，该类具有两个方法：__init__和__repr__。__init__方法是类的构造函数，用于初始化对象的属性。接受一个参数 strData，并将其赋值给对象的 Data 属性。如果没有提供参数，则默认为空字符串。__repr__方法是一个特殊的方法，用于返回对象的字符串表示形式。创建了一个 strReverse 对象，并将字符

串"ABC"作为参数传递给构造函数。然后，打印该对象的字符串表示形式，即逆序后的字符串"CBA"。print( )函数是 Python 中的一个内置函数。故正确答案为 B 选项。

6.有关下面 Python 代码的说法，正确的是（）。

```
class Num:
    def __init__(self,val):
        self.Value = val
    def __add__(self,other):
        return self.Value +
other.Value
    def add(self,other):
        return self.Value +
other.Value

a = Num(10)
print(a + Num(20),
Num(20).__add__(a), a.add(Num(20)))
print(a)
```

A.在倒数第 2 行代码中， a + Num(20) 将执行正确，而 Num(20).__add__(a) 将导致错误

B.由于类Num中没有定义加号运算符，所以倒数第2行代码中的 a + Num(20) 被执行时将导致错误

C.如果将倒数第 2 行代码中的 a.add(num(20)) 修改为 Num(20).add(a) 将导致错误，因为 Num(20)不是一个对象，而a 是类 Num 的对象

D.倒数第 1 行代码 print(a) 将被正确执行，虽然没有定义相关成员函数，或者称之为方法

【答案】D

【解析】本题属于考察 Python 类的创建知识。实例化对象后可以直接输出，不需要将其赋值给变量。因为 Python 是一种动态类型语言，它允许我们在运行时创建和操作对象，而无需事先声明变量的类型。当我们实例化一个对象时，Python 会自动为其分配内存空间，并将其引用存储在当前的命名

空间中。因此，我们可以直接使用对象进行操作，而无需将其赋值给变量。当我们创建一个类的实例时，可以通过 print()函数将其输出。输出的结果将显示对象的内存地址（即 id），__add__(self, other): 特殊方法，用于实现加法运算符（+）的行为。当使用加法运算符将两个 Num 对象相加时，会调用该方法。它接受一个参数 other，表示另一个 Num 对象，并返回两个对象的 Value 属性之和。add(self, other):与特殊方法__add__功能相同。故正确答案为 D 选项。

7.有关下面 Python 代码的说法，正确的是（）。

```
class manyData:
    def __init__(self, lstData):
        self.__data = lstData
    def push(self, val):
        self.__data.append(val)
        return self
    def pop(self):
        popVal = self.__data[-1]
        self.__data.pop()
        return popVal
    def __len__(self):
        return len(self.__data)

myData = manyData([1,2,3])
myData.push(100)
print(len(myData))
print(myData.peek())
print(myData.pop())
```

A.manyData 类可用于构造队列(queue)数据结构

B.在上面代码环境，代码 myData.__data.append(10) 可以增加 10 到 myData.__data 之中

C.len()是 Python 内置函数，不适用于上面代码环境中的 manyData

D.异常处理可以用于自定义类，因此 manyData 类的 pop()函数执行可以增加异常处理代码，否则可能导致异常

【答案】D

【解析】本题属于考察 Python 类实现栈的知识。manyData 类实现了构造了一种“先进先出，后进后出”的数据结构，符合栈的特点。在类的定义中，使用了双下划线前缀来表示私有属性（如__data），用于指示这些属性应该为私有的，并且不应该直接访问或修改它们。如果列表为空，调用 pop() 会抛出 IndexError 异常。在使用 pop() 之前，使用异常处理代码先检查列表是否为空。故正确答案为 D 选项。

8.有关下面 Python 代码的说法，错误的是（）。

```
class moreData:
    def __init__(self, lstData):
        self.__data = lstData
    def push(self, val):
        self.__data.append(val)
        return self
    def pop(self):
        popVal = self.__data[0]
        self.__data.pop(0)
        return popVal
    def __len__(self):
        return len(self.__data)

myData = moreData([1,2,3])
myData.push(11).push(12).push(13)
print(myData.pop())
```

A. moreData 类可用于构造队列(queue)数据结构

B. 代码倒数第 2 行连续 push()用法将导致错误

C. moreData 可以认为是 list 类型的适配器，裁剪了 list 功能

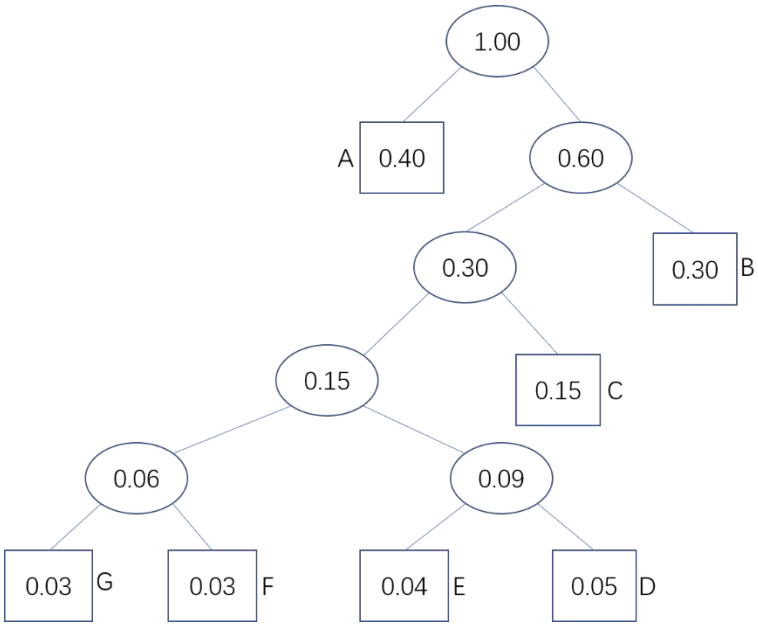
D. data 可以认为是 moreData 类的私有成员，只能在类内访问

【答案】B

【解析】本题属于考察 Python 类实现队列的知识。moreData 类实现了构造了一种“先进后出，后进先出”的数据结构，符合队列的特点。类的实例化对象可以多次调用其所属类的方法，只要方法内部没有出现逻辑错误或者异常情况。类的定义中，使用了双下划线前缀来表示私有属性（如__data），用于指示这些属性应该为私有的，不可以直接访问或修改它们。Python 适配器可以帮助我们使用旧

的代码，在不必要改变太多的情况下，将其与新的框架和库相结合。故正确答案为 B 选项。

9.某内容仅会出现 ABCDEFG，其对应的出现概率为 0.40、0.30、0.15、0.05、0.04、0.03、0.03，如下图所示。按照哈夫曼编码规则，假设 B 的编码为 11，则 D 的编码为( )。



A. 10010

B. 10011

C. 10111

D. 10001

【答案】B

【解析】本题属于考察哈夫曼编码和哈夫曼树的知识。哈夫曼编码的主要思想是根据字符出现的频率来构建一棵哈夫曼树，使得频率高的字符具有较短的编码，从而减少数据的存储空间和传输时间。从哈夫曼树的根节点开始，向左走标记为 0，向右走标记为 1，直到叶子节点结束。将路径上的标记连接起来就得到了该叶子节点对应字符的哈夫曼编码。所以 D 的哈夫曼编码就是 10011 故正确答案为 B 选项。

10.下面有关格雷码的说法，错误的是 ( )。

A. 在格雷码中，任意两个相邻的代码只有一位二进制数不同

B. 格雷码是一种唯一性编码

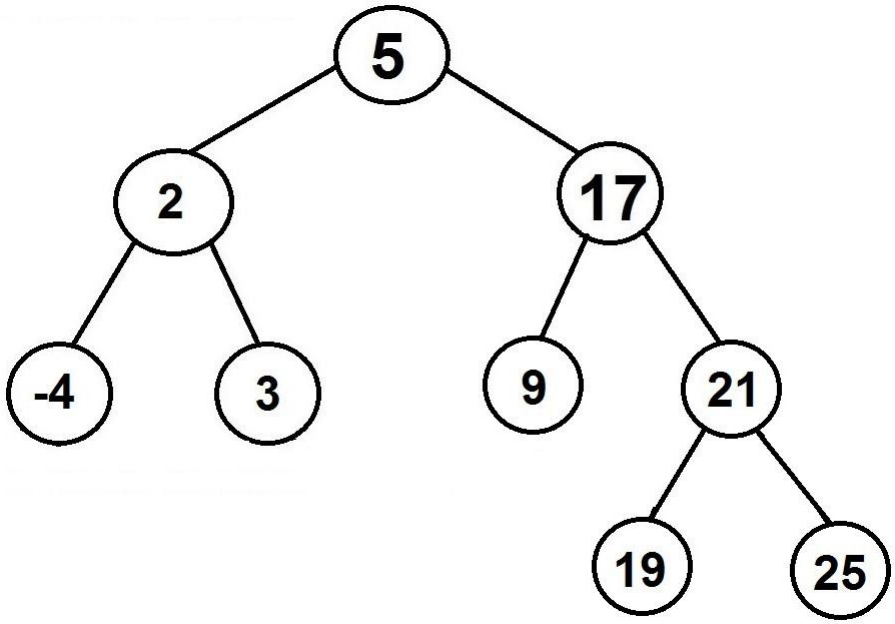
C. 在格雷码中，最大数和最小数只有一位二进制数不同

D. 格雷码是一种可靠性编码

【答案】B

【解析】本题属于考察格雷码知识。格雷码，也叫二进制格雷码，特点是在一个数的编码中，任意两个相邻的代码之间仅有一位二进制数不同，同时，最大数与最小数之间也仅一位数不同，即“首尾相连”。此外，格雷码是一种具有反射特性和循环特性的单步自补码。其循环和单步特性消除了随机取数时出现重大错误的可能，其反射和自补特性使得对其进行求反操作也非常方便。因此，格雷码属于一种可靠性编码，是一种错误最小化的编码方式。但是，格雷码的顺序并不是唯一的，有多种方式可以生成同一个格雷码。例如，长度为 3 的普通二进制码有 8 种可能：000、001、010、011、100、101、110、111；而长度为 3 的格雷码也有 8 种可能：000、001、011、010、110、111、101、100 故正确答案为 B 选项。

11.有关下图的二叉树，说法正确的是（ ）。



A. 既是完全二叉树也是满二叉树

B. 既是二叉搜索树也是平衡二叉树

C. 非平衡二叉树

D. 以上说法都不正确

【答案】B

【解析】本题属于考察树的定义，构造和完全二叉树知识。完全二叉树、满二叉树、二叉搜索树和平衡二叉树，这四种都是常见的二叉树结构，它们之间有显著的区别。首先，完全二叉树与满二叉树是一类特殊的二叉树。完全二叉树是由满二叉树转化而来的，具体的转化方法是将满二叉树从最后一个节点开始删除，一个一个从后往前删除，剩下的就是完全二叉树，最后一层的所有结点都必须连续集中在树的最左边。而满二叉树是指如果一个二叉树的层数为 K，且结点总数是 (2^k) -1 ，则它就是满二叉树。所以题目中的二叉树不满足要求， 不是满二叉树或完成二叉树。其次，二叉搜索树对于任何的一个结点，其左子树上所有结点的值都小于该结点的值，右子树上所有结点的值都大于该结点的值。最后，平衡二叉树则是一种特殊的二叉搜索树。平衡二叉树特点是左右两个子树的高度差不超过 1，且左右两个子树都是一棵平衡二叉树。故正确答案为 B 选项。

12.N 个节点的二叉搜索树，其查找的平均时间复杂度为（ ）。

A.O(1)

B.O(N)

C.O($\log N$)

D.O(N^2)

【答案】C

【解析】本题属于考察二叉树的知识。二叉搜索树对于任何节点，其左子树上所有节点的值都小于该节点，而右子树上所有节点的值都大于该节点，可以根据关键字与当前节点的比较结果，决定是在左子树还是右子树中继续进行查找，从而将查找范围每次都缩小一半。对于 n 个节点的二叉搜索树来说，其高度为[log2(n)]+1，所以查找的平均时间复杂度为 O(logn)。故正确答案为 C 选项。

13.青蛙每次能调 1 或 2 步。下面是青蛙跳到第 N 步台阶 Python 实现代码。该段代码采用的算法是（ ）。

```
def jumpFrog(N):
    if N <= 3:
        return N
    else:
        return jumpFrog(N - 1) +
jumpFrog(N - 2)
print(jumpFrog(4))
```

A. 递推算法

B. 贪心算法

C. 动态规划算法

D. 分治算法

【答案】C

【解析】本题属于考察简单动态规划算法的知识，可以通过动态规划来完成求解完成跳台阶方法。故正确答案为 C 选项。

14.Python 字典值查找的时间复杂度是（ ）。

A.O(1)

B.O(N)

C.O($\log N$)

D.O(N^2)

【答案】A

【解析】本题属于考察 Python 字典时间复杂度的知识。Python 字典值查找的时间复杂度是 O(1)，是因为在 Python 中，字典是通过哈希表实现的。哈希表是一种数据结构，它使用一个哈希函数将键映射到数组的一个位置上，因此可以通过键直接访问对应的值。当进行查找操作时，Python 会先计算键的哈希值，然后在哈希表中查找该位置是否存在对应的键值对。由于哈希函数的设计和哈希表的实现方式，这个查找过程的时间复杂度是常数级别的，即 O(1)。故正确答案为 A 选项。

15.下面有关 Python 的 in 运算符说法错误的是（ ）。

A. 对于不同的数据类型，in 运算符的时间复杂度不同

B. 对于 set 和 dict 类型，in 运算符的时间复杂度是 O(1)

C. 对于 list 和 tuple 类型，in 运算符的时间复杂度是 O(N)

D. 对于 Python 的 in 运算符，其时间复杂度相同

【答案】D

【解析】本题属于考察 Python 语言 in 运算符的时间复杂度问题。对于列表、元组和字符串等可迭代对象，in 运算符的时间复杂度为 O(n)，其中 n 为可迭代对象的长度。这是因为在最坏的情况下，需要遍历整个可迭代对象才能找到目标元素。对于集合和字典等无序容器，in 运算符的时间复杂度为 O(1)。这是因为集合和字典内部使用哈希表实现，可以直接通过键值快速查找到对应的元素。对于不可变的数据类型（如整数、浮点数和布尔值）的对象是预先在内存中存储的，每个对象的值都是固定的，所以可以通过直接访问内存地址来获取其值，而不需要遍历任何数据结构，所以 in 运算符的时间复杂度也是 O(1)。故正确答案为 D 选项。

二、判断题（每题 2 分，共 20 分）

1. TCP/IP 的传输层的两个不同的协议分别是 UDP 和 TCP。

【答案】正确√

【解析】本题考察计算机网络的基本知识。TCP/IP 的传输层有两个主要的不同协议：TCP（传输控制协议）和 UDP（用户数据报协议）。TCP 协议是一个面向连接的、可靠的传输协议，当数据通过 TCP 协议进行传输时，发送方和接收方之间的通信会建立一个虚拟连接，然后按照这个连接有序地传输数据，确保数据的完整性和正确性。UDP 协议则是一种无连接的、不可靠的传输协议，它只是简单地将数据包发送出去，而不保证数据能否到达目的地。这种协议适用于那些对数据传输的可靠性要求不高，但要求速度快的场景，例如视频直播、在线游戏等。所以本题正确。

2. 5G 网络中，5G 中的 G 表示 Gigabytes/s，其中 1 GB = 1024 MB。

【答案】错误×

【解析】本题考察计算机网络的知识。5G 中的 G 表示 Generation，即“代”的意思。5G 是第五代移动通信技术的简称，意味着超快的数据传输速度。它的理论峰值可达每秒数十 Gb，比 4G 网络的传输速度快数百倍，你可以在 1 秒之内完成整部超高画质电影的下载。随着 5G 技术的诞生，用智能终端分享 3D 电影、游戏以及超高画质（UHD）节目的时代即将到来。所以本题错误。

3. 在面向对象中，类是对象的实例。

【答案】错误×

【解析】本题考察 Python 类和类的实例化的知识。在面向对象编程中，类是一种抽象的数据类型，对象则是类的实例化，即根据类的定义所创建的具体实体。例如，我们可以定义一个“汽车”类，包含属性如颜色、品牌、型号等，以及方法如启动、停止、加速等。然后，我们可以根据这个类创建一个具体的对象，比如一辆红色的奔驰车，它具有自己的颜色、品牌和型号属性，以及启动、停止和加速等方法的实现。因此，可以说类是对象的模板或蓝图，而对象则是类的实例化结果。所以本题错误。

4. 在 Python 类的定义中，可以有类属性和实例属性，类属性值被该类的对象共享。

【答案】正确√

【解析】本题考察 Python 类的创建中类属性和实例属性的知识。类属性是指定义在类中的变量，它们共享于该类的所有实例对象。类属性可以通过类名或实例对象来访问。类属性在类定义时就已经存在，并且在整个程序运行期间都不会被改变。实例属性是指定义在类的__init__方法中的变量，它们是每个实例对象所独有的。实例属性只能通过实例对象来访问，而不能通过类名直接访问。实例属性的值在创建实例对象时被初始化，并且可以在程序运行期间被修改。所以本题正确。

5. 在 Python 类的定义中，可以用魔法方法定义初始化函数或运算符函数等。【答案】正确√

【解析】本题考察 Python 类的创建知识。魔术方法的名称以双下划线开头和结尾，并且第一个参数通常是 self，表示当前对象自身。通过重写这些方法，可以实现自定义的行为和功能。所以本题正确。

6. DFS 是深度优先搜索算法的英文简写。

【答案】正确√

【解析】本题考察深度优先搜索算法的概念。深度优先搜索（Depth First Search）简称深搜或者 DFS，是一种用于遍历或搜索树或图的算法。这种算法既适用于无向图（网），也适用于有向图（网）。它的基本思想是对每一个可能的分支路径深入到不能再深入为止，而且每个顶点只访问一次。所以本题正确。

7. 哈夫曼编码是一种有损压缩算法。

【答案】错误×

【解析】本题考察哈夫曼编码的知识。哈夫曼编码是一种用于无损数据压缩的算法，而不是有损压缩。它通过为每个字符分配唯一的二进制编码来实现压缩。这种编码方式使得出现频率较高的字符具有较短的编码，而出现频率较低的字符具有较长的编码，从而减少了数据的总长度。所以本题错误。

8．Python 本身并不支持指针和引用语法，因此有关链表等算法或数据结构在 Python 中不能实现。

【答案】错误×

【解析】本题考察链表的知识。虽然 Python 中没有像 C 语言中的显式指针类型，但是 Python 中的对象的引用可以类比为指针。在 Python 中，变量只是一个对对象的引用，而不是对象本身。当我们对变量进行赋值时，实际上是将一个对象的

引用赋值给变量。因此，尽管 Python 中没有明确的“指针”定义，我们仍然可以在其内部实现类似于指针的操作。至于链表这种数据结构，它是一种在存储单元上非连续、非顺序的存储结构，和数组相比，它并不需要一块连续的内存空间，而是通过“指针”将一组零散的内存块串联起来使用。所以，我们可以在 Python 中实现链表等算法或数据结构。例如，每个链表节点包含两部分：数据域与指针域，其中指针域就是保存下一个节点信息的“指针”。总的来说，虽然 Python 中没有直接的指针语法支持，但借助其对象的引用特性以及一些内置的数据结构（如列表），我们依然可以实现链表等复杂的数据结构和算法。所以本题错误。

9. 如果节点数为 N，广度搜索算法的最差时间复杂度为 O(N)。

【答案】正确√

【解析】本题考察广度搜索算法的知识。广度优先搜索（BFS）算法是一种用于遍历或搜索树或图的算法。在最坏的情况下，广度优先搜索算法的时间复杂度为 O(N)，其中 N 是节点的数量。这是因为广度优先搜索算法需要访问图中的所有节点，并且每个节点都需要被访问一次。因此，如果图中有 N 个节点，那么广度优先搜索算法就需要进行 N 次操作。所以本题正确。

10. 二叉搜索树的左右子树也是二叉搜索树。

【答案】正确√

【解析】本题属于二叉搜索树的知识。二叉搜索树对于任何的一个结点，其左子树上所有结点的值都小于该结点的值，右子树上所有结点的值都大于该结点的值，所以本题正确。

三、编程题（每题 25 分，共 50 分）

1.小杨买饮料

【问题描述】

小杨来到了一家商店，打算购买一些饮料。这家商店总共出售 N 种饮料，编号从 0 至 N-1，其中编号为 i 的饮料售价c_i元，容量l_i毫升。

小杨的需求有如下几点：

1. 小杨想要尽可能尝试不同种类的饮料，因此他希望每种饮料至多购买 1 瓶；
2. 小杨很渴，所以他想要购买总容量不低于 L 的饮料；
3. 小杨勤俭节约，所以在 1 和 2 的前提下，他希望使用尽可能少的费用。

方便起见，你只需要输出最少花费的费用即可。特别地，如果不能满足小杨的要求，则输出 no solution。

【输入描述】

第一行两个整数 N,L。

接下来 N 行，依次描述第 i=0，1，.....N-1 种饮料：每行两个整数c_i, l_i。

【输出描述】

输出一行一个整数，表示最少需要花费多少钱，才能满足小杨的要求。特别地，如果不能满足要求，则输出 no solution 。

【样例输入 1】

```
5 100
100 2000
2 50
4 40
5 30
3 20
```

【样例输出 1】

```
9
```

【样例解释 1】

小杨可以购买 1,2,4 号饮料，总计获得 50+40+20=110 毫升饮料，花费 2+4+3=9 元。

如果只考虑前两项需求，小杨也可以购买 1，3，4 号饮料，它们的容量总和为 50+30+20=100 毫升，恰好可以满足需求。但遗憾的是，这个方案需要花费 2+5+3=10 元。

【样例输入 2】

5 141

100 2000

2 50

4 40

5 30

3 20

【样例输出 2】

100

【样例解释 2】

1，2，3，4 号饮料总计 140 毫升，如每种饮料至多购买 1 瓶，则恰好无法满足需求，因此只能花费 100 元购买 0 号饮料。

【样例输入 3】

4 141

2 50

4 40

5 30

3 20

【样例输出 3】

no solution

【数据规模】

对于 40%的测试点，保证 $N \leq 20$；$1 \leq L \leq 100$；$l_i \leq 100$。

对于 70%的测试点，保证$l_i \leq 100$。

对于所有测试点，保证 $1 \le N \le 500$；$1 \le L \le 2000$；$1 \le c_i$，$l_i \le 100$。

【题目大意】

小杨要用尽可能少的费用从 N 种饮料中购买总容量不低于 L 的饮料，每个饮料都给出对应的售价和容量。

【解题思路】

本题主要考察动态规划算法，是背包问题的变体。

1. 输入两个整数 n 和 L，然后创建一个长度为 L 的列表 dp，初始化为 10 的 10 次方，但 dp[0]被设置为 0。

2. 进行 n 次循环，每次循环中，读取两个整数 c 和 l，然后更新 dp 列表。

3. 最后，如果 ans 仍然是无穷大，那么输出"no solution"，否则输出 ans

【参考程序】

```
n, L = map(int, input().split())
inf = 10 ** 10
dp = [inf for i in range(L)]
dp[0] = 0
ans = inf
for i in range(n):
    c, l = map(int, input().split())
    for j in range(max(L - l, 0), L):
        ans = min(ans, dp[j] + c)
    for j in range(L - 1, l - 1, -1):
        dp[j] = min(dp[j], dp[j - l] + c)
if ans == inf:
    print("no solution")
else:
    print(ans)
```

2. 小杨的握手问题

【问题描述】

小杨的班级里共有 N 名同学，学号从 0 至 N-1。

某节课上，老师安排全班同学进行一次握手游戏，具体规则如下：老师安排了一个顺序，让全班 N 名同学依次进入教室。每位同学进入教室时，需要和**已经在教室内**且**学号小于自己**的同学握手。

现在，小杨想知道，整个班级总共会进行多少次握手。

提示：可以考虑使用归并排序进行降序排序，并在此过程中求解。

【输入描述】

输入包含 2 行。第一行一个整数 N，表示同学的个数；第二行 N 个用单个空格隔开的整数，依次描述同学们进入教室的顺序，每个整数在 0~N-1 之间，表示该同学的学号。

保证每位同学会且只会进入教室一次。

【输出描述】

输出一行一个整数，表示全班握手的总次数。

【样例输入 1】

4

2 1 3 0

【样例输出 1】

2

【样例解释 1】

2 号同学进入教室，此时教室里没有其他同学。

1 号同学进入教室，此时教室里有 2 号同学。1 号同学的学号小于 2 号同学，因此他们之间不需要握手。

3 号同学进入教室，此时教室里有 1,2 号同学。3 号同学的学号比他们都大，因此 3 号同学需要分别和另外两位同学握手。

0 号同学进入教室，此时教室里有 1,2,3 号同学。0 号同学的学号比他们都小，因此 0 号同学不需要与其他同学握手。

综上所述全班一共握手 0+0+2+0=2 次。

【样例输入 2】

6

0 1 2 3 4 5

【样例输出 2】

15

【样例解释 2】

全班所有同学之间都会进行握手，因为每位同学来到教室时，都会发现他的学号是当前教室里最大的，所以他需要和教室里的每位其他同学进行握手。

【数据规模】

对于 30%的测试点，保证 $N \leq 100$。

对于所有测试点，保证 $2 \leq N \leq 3 \times 10^5$。

【题目大意】

全班 N 名同学依次进入教室。每位同学进入教室时，需要和已经在教室内且学号小于自己的同学握手，求整个班级总共会进行多少次握手。可以考虑使用归并排序进行降序排序

【解题思路】

本题主要考察归并排序算法的实现。函数 merge(arr, l, r)接受一个数组 arr 和两个索引 l 和 r 作为参数，表示要排序的数组范围。函数返回一个整数，表示将数组从索引 l 到 r 范围内的元素排序所需的最小交换次数。

1. 首先检查数组长度是否为 2，如果是，则不需要进行排序，直接返回 0。
2. 然后计算中间索引 mid，并将问题分为两个子问题：左半部分和右半部分。递归调用 merge(arr, l, mid)和 merge(arr, mid, r)分别计算左右子问题的最小交换次数，并将它们相加得到总的交换次数 ans。
3. 创建一个空列表_arr，用于存储合并后的有序数组。使用两个指针 i 和 j 分别指向左半部分和右半部分的第一个元素。
4. 通过比较两个指针所指向的元素的大小，将较小的元素添加到_arr 中，并更新相应的指针。

5. 如果右半部分的元素较小，还需要将左半部分剩余的元素全部添加到_arr 中，并更新指针。累加交换次数 ans，将右半部分剩余的元素移动到正确的位置。
6. 最后，将_arr 中的元素复制回原数组 arr 的相应位置，并返回总的交换次数 ans。
7. 获取输入的数据 n，arr。
8. 输出调用 merge(arr，0，n)函数后的返回值。

【参考程序】

```
def merge(arr, l, r):
    if l + 1 == r:
        return 0
    mid = (l + r) >> 1
    ans = merge(arr, l, mid) + merge(arr, mid, r)
    _arr = []
    i, j = l, mid

    for _ in range(l, r):
        if j == r or (i < mid and arr[i] > arr[j]):
            _arr.append(arr[i])
            i += 1
        else:
            _arr.append(arr[j])
            j += 1
            ans += mid - i
    for idx, item in enumerate(_arr):
        arr[l + idx] = item
    return ans


n = int(input())
arr = list(map(int, input().split(' ')))
ans = merge(arr, 0, n)
print(ans)
```