2023 年 GESP9 月认证 Python 五级试卷解析

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 五级认证真题解析。

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 Python 语言特点的基本知识。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.下列 Python 代码执行后输出的是（ ）。

```
lstA = [1,2,3]
lstB = [4,5,6,lstA]
del lstA
print(lstB)
```

A. [4, 5, 6, lstA]

B. [4, 5, 6, 1, 2, 3]

C. [4, 5, 6, [1, 2, 3]]

D. 执行将报错，因为 lstA 已经被删除

【答案】C

【解析】本题属于考察 Python 列表中嵌套列表的知识。定义了 lstA 和 lstB 两个列表，然后将 lstA 赋值给 lstB，实现了嵌套列表的效果。最后，打印出 lstB 的内容，它包含了四个元素，其中第四个元素是嵌套的 lstA 列表对应的对象。由于在 Python 中采用了引用计数机制，每个对象都有一个引用计数的属性，记录着有多少个变量指向该对象。当一个对象的引用计数为 0 时，它将被垃圾回收机制清除由于 lstB 引用了 lstA，即数据[1, 2, 3]被引用了两次，del lstA 仅删除一次，[1, 2, 3]将继续存在。故正确答案为 C 选项。

4.有关下面 Python 代码说法错误的是（）。

```
#sumA()和sumB()用于求1~N之和
def sumA(N):
    ret = 0
    for i in range(1, N + 1):
        ret += i
    return ret

def sumB(N):
    if N == 1:
        return 1
    else:
        return N + sumB(N - 1)

N = int(input("请输入大于等于1的正整
数:"))
print(sumA(N),sumB(N))
```

A.sumA() 用循环方式求 1~N 之和， sumB()用递归方式求 1~N 之和

B.默认情况下，倒数第二行被执行时如果输入较小的正整数如 100 ，即倒数第一行能正确被执行，则能实现求到 1 到 N 之和

C.默认情况下，倒数第二行被执行时如果输入较大的正整数如 10000 ，倒数第一行被能正确被执行，能实现求到 1 到 N 之和

D.默认情况下，一般说来，sumA() 和效率高于 sumB()

【答案】C

【解析】本题属于考察递归算法的知识。递归调用的次数必须是有限的且递归必定有结束条件来终止递归，所以 sumB()用递归方式求 1~N 之和。同时在 Python 中，当递归调用的层数过多，超过了 Python 默认的限制（通常为 1000），就会触发"maximum recursion depth exceeded in comparison"错误。故正确答案为 C 选项。

5.下面 Python 代码以递归方式现字符串反序，横线处应填上代码是（）。

```
#字符串反序
def sReverse(sIn):
    if len(sIn) <= 1:
        return sIn
    else:
        return _____________________

print(sReverse("Hello"))
```

A. sReverse(sIn[1:]) + sIn[:1]

B. sReverse(sIn[:1]) + sIn[1:]

C. sIn[:1] + sReverse(sIn[1:])

D. sIn[1:] + sReverse(sIn[:1])

【答案】A

【解析】本题属于考察递归算法和字符串切片的知识。切片操作使用冒号分隔起始和结束索引，格式为：str[start:end]，其中 start 是切片开始的位置，end 是切片结束的位置（不包含）。如果省略 start，则默认从字符串的开头开始；如果省略 end，则默认一直切到字符串的末尾。sReverse 函数的作用是将输入的字符串 sIn 进行反转。如果输入的字符串长度小于等于 1，则直接返回该字符串；否则，将字符串的第一个字符移到最后，并递归调用 sReverse 函数处理剩余的子串，从而实现字符串的反序输出。故正确答案为 A 选项。

6.印度古老传说：创世时有三根金刚柱，其中一柱从下往上按照大小顺序摞着 64 片黄金圆盘，当圆盘逐一从一柱借助另外一柱全部移动到另外一柱时，宇宙毁灭。移动规则：在小圆盘上不能放大圆盘，在三根柱子之间一次只能移动一个圆盘。下面的 Python 代码以递归方式实现汉诺塔，横线处应填入代码是( )。

```
#递归实现汉诺塔，将N个圆盘从A通过B移动C
#圆盘从底到顶，半径必须从大到小
def Hanoi(A, B, C, N):
    if N == 1:
        print(A, "->", C)
    else:
        Hanoi(A, C, B, N-1)
        print(A, "->", C)
        ____________________________

Hanoi("甲", "乙", "丙" , 3)
```

A. Hanoi(B,C,A,N-2)

B. Hanoi(B,A,C,N-1)

C. Hanoi(A,B,C,N-2)

D. Hanoi(C,B,A,N-1)

【答案】B

【解析】本题属于考察递归算法的知识。汉诺塔问题是一个典型的递归问题，递归的基本思想是将问题分解为规模更小的子问题。在这个问题中，我们可以将 n-1 个圆盘的移动过程看作是一个子问题，然后将第 n 个圆盘移动到目标柱子上。递归函数的终止条件是当只有一个圆盘时，直接将其移动到目标柱子上。递归调用时，需要注意移动的顺序，先将 n-1 个圆盘从源柱子移动到辅助柱子，然后将第 n 个圆盘从源柱子移动到目标柱子，最后将 n-1 个圆盘从辅助柱子移动到目标柱子。本题辅助柱子是 B，目标柱子是 C。故正确答案为 B 选项。

7.根据下面 Python 代码的注释，横线处应填入( )。

```
isOdd = lambda N: N % 2 == 1

lstA = list(range(1,100))
lstA.sort(key = ____________)#lstA成员
偶数在前，奇数在后
lstB = [x for x in lstA if _________]
#lstB成员全为奇数
print(lstA, lstB)
```

A. isOdd isOdd(x)

B. isOdd(x) isOdd

C. isOdd isOdd

D. isOdd(x) isOdd(x)

【答案】A

【解析】本题属于考察 Python 中 sort()函数，lamdba 函数与列表推导式结合使用的知识。sort()函数的 key 参数可以接受一个函数作为值。当 key 值为函数名不带括号时，表示将列表中的每个元素作为参数传递给该函数，并将函数的返回值作为排序依据。这样做的好处是可以根据自定义的函数逻辑对列表进行排序。所以第一个横线为 isOdd，第二个横线处为列表推导式，要使用函数，需要为 isOdd（x）。故正确答案为 A 选项。

8.有关下面 Python 代码正确的是（ ）。

```
def isEven(N):
    return N % 2 == 0

def checkNum(Fx,N):
    return Fx(N)

print(checkNum(isEven,10))
```

A. checkNum() 函数定义错误

B. 最后一行代码将 isEven 作为 checkNum() 参数将导致错误

C. 最后一行代码执行后将输出 True

D. 触发异常

【答案】C

【解析】本题属于考察 Python 高阶函数知识。在 Python 中，函数名可以作为函数参数传递。这意味着可以将一个函数作为另一个函数的参数，从而实现更高级的编程功能。这种特性被称为高阶函数。通过将函数名作为参数传递，可以实现更加灵活和可扩展的代码结构。故正确答案为 C 选项。

9.有关下面 Python 代码正确的是（ ）。

```
isOdd = lambda N: N % 2 == 1

def Add(N,M):
    return N+M

def checkNum(Fx):
    return Fx

print(checkNum(isOdd)(10))
print(checkNum(Add)(10,20))
```

A. checkNum() 函数定义错误

B. 倒数第 2 行代码将 isOdd 作为 checkNum() 参数将导致错误

C. 最后一行代码将 Add 作为 checkNum() 参数将导致错误

D. 倒数两行代码执行后都将没有错误

【答案】D

【解析】本题属于考察 Python 函数的知识。函数调用双括号可以用来表示函数参数的传递。这意味着我们可以直接使用函数的返回值作为参数传递给另一个函数。故正确答案为 D 选项。

10.下面代码执行后的输出是（ ）。

```
def jumpFloor(N):
    print(N, end = "#")
    if N == 1 or N == 2:
        return N
    else:
        return jumpFloor(N-1) +
jumpFloor(N-2)

print(jumpFloor(4))
```

A. 4#3#2#2#4

B. 4#3#2#2#1#5

C. 4#3#2#1#2#4

D. 4#3#2#1#2#5

【答案】D

【解析】本题属于考察递归算法的知识。这段代码是一个递归函数，用于解决经典的“跳台阶问题”。问题描述如下：一只青蛙在 N 阶台阶上，每次可以跳 1 阶或 2 阶，问有多少种不同的跳法。函数名为 jumpFloor，参数为 N，表示台阶的阶数。函数内部首先打印出当前台阶数 N，然后判断是否为 1 阶或 2 阶，如果是则直接返回该台阶数作为跳法数量。否则，通过递归调用自身来计算跳法数量，即 jumpFloor(N-1)表示跳 1 阶的跳法数量，jumpFloor(N-2)表示跳 2 阶的跳法数量。最后将两者相加得到总的跳法数量并返回。故正确答案为 D 选项。

11.下面 Python 代码中的 isPrimeA() 和 isPrimeB() 都用于判断参数 是否为素数，有关其时间复杂度的正确说法是（ ）。

```
def isPrimeA(N):
    if N < 2:
        return False

    for i in range(2,N):
        if N % i == 0:
            return False
    return True

def isPrimeB(N):
    if N < 2:
        return False

    endNum = int(N ** 0.5)
    for i in range(2,endNum+1):
        if N % i == 0:
            return False
    return True

print(isPrimeA(13),isPrimeB(13))
```

A.isPrimeA()的最坏时间复杂度是 O(N)，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(N)，isPrimeB()的最坏时间复杂度是 $O(N^{\frac{1}{2}})$，isPrimeB()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)，isPrimeB()优于 isPrimeA()。

【答案】B

【解析】本题属于唯一分解定理、素数表的线性筛法和算法复杂度的估算综合应用的知识。唯一分解定理是数论中的一个重要定理，对于任意一个大于 1 的整数 n，存在唯一的一对素数 p 和 q，使得 n = p * q。时间复杂度越大,算法的执行效率越低。$O(N^{\frac{1}{2}}) < O(N)$，所以 isPrimeB()优于 isPrimeA()。故正确答案为 B 选项。

12.下面 Python 代码用于归并排序，其中 merge() 函数被调用次数为（ ）。

```
def mergeSort(listData):
    if len(listData) <= 1:
        return listData

    Middle = len(listData) // 2
    Left, Right = mergeSort(listData[:Middle]),
mergeSort(listData[Middle:])

    return merge(Left, Right)
```

```
def merge(Left, Right):
    Result = []
    i, j = 0 , 0

    while i < len(Left) and j < len(Right):
        if Left[i] <= Right[j]:
            Result.append(Left[i])
            i += 1
        else:
            Result.append(Right[j])
            j += 1

    Result.extend(Left[i:])
    Result.extend(Right[j:])

    return Result

lstA = [1, 3, 2, 7, 11, 5, 3]
lstA = mergeSort(lstA)

print(lstA)
```

A. 0

B. 1

C. 6

D. 7

【答案】C

【解析】本题属于考察归并排序知识。归并排序是一种分治算法，它将待排序的序列分为两个子序列，对每个子序列进行递归排序，然后将两个有序的子序列合并成一个有序的序列。根据题目描述，代码中定义了两个函数：mergeSort( ) 和 merge( )。mergeSort( )是归并排序的主函数，它接受一个列表作为参数，并返回排好序的列表。如果列表的长度小于等于 1，直接返回该列表；否则，将列表分成左右两个子列表，分别对它们进行归并排序，然后将排好序的子列表通过 merge 函数合并成最终的结果。merge( )函数用于合并两个已排序的子列表。它接受左右两个子列表作为参数，并返回一个新的有序列表。在合并过程中，使用 i 和 j 分别指向左右子列表的起始位置，比较左右子列表的元素大小，将较小的元素添加到结果列表中，并将对应的值加 1，即向后移动一位。当其中一个子列表遍历完后，将另一个子列表剩余的元素添加到结果列表中。题目中最后会得到

一个列表[1]，此时列表长度小于等于 1，直接返回列表。故正确答案为 C 选项。

13.在上题的归并排序算法中，代码 Left, Right = mergeSort(listData[:Middle]), mergeSort(listData[Middle:]) 涉及到的算法有（ ）。

A. 搜索算法

B. 分治算法

C. 贪心算法

D. 递推算法

【答案】B

【解析】本题属于考察归并排序知识。归并排序是一种分治算法，它将待排序的序列分为两个子序列，对每个子序列进行递归排序，然后将两个有序的子序列合并成一个有序的序列。故正确答案为 B 选项。

14.归并排序算法的基本思想是（ ）。

A. 将数组分成两个子数组，分别排序后再合并。

B. 随机选择一个元素作为枢轴，将数组划分为两个部分。

C. 从数组的最后一个元素开始，依次与前一个元素比较并交换位置。

D. 比较相邻的两个元素，如果顺序错误就交换位置。

【答案】A

【解析】本题属于考察归并排序知识。归并排序是一种分治算法，它将待排序的序列分为两个子序列，对每个子序列进行递归排序，然后将两个有序的子序列合并成一个有序的序列。故正确答案为 A 选项。

15.有关下面 Python 代码的说法正确的是（ ）。

```
class Node:
    def __init__(self, Val, Nxt = None):
        self.Value = Val
        self.Next = Nxt

firstNode = Node(10)
firstNode.Next = Node(100)
firstNode.Next.Next = Node(111,firstNode)
```

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

【答案】C

【解析】本题属于考察 Python 实现链表知识。循环链表是一种特殊的链表结构，它的最后一个节点指向链表的第一个节点，形成一个环。。故正确答案为 C 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. 在特殊情况下流程图中可以出现三角框和圆形框。

【答案】错误×

【解析】如果流程图需要表达一个循环结构，可以使用圆形框来表示循环的开始和结束。如果流程图需要表达一个并行或并发结构，可以使用多个圆形框或菱形框来表示不同的分支或任务。但没有三角框，所以本题错误。

3. 找出自然数 N 以内的所有质数常用埃氏筛法，其时间复杂度为 O(N)。

【答案】错误×

【解析】本题考察素数表的埃氏筛法的知识。埃氏筛法是一种用于求解小于等于给定数 N 的所有质数的算法。它的基本思想是从 2 开始，将 2 的倍数筛掉，然后找到下一个未被筛掉的数，将其倍数筛掉，如此循环，直到筛完所有小于等于 N 的数。最后剩下的就是质数。埃氏筛法的时间复杂度为 O(n log log n),所以本题错误。

4. Python 的 in 运算符相当于通过查找算法判断元素是否存在，其查找通常采用二分法。

【答案】错误×

【解析】本题考察二分查找法的知识。二分查找法的工作原理是每次将数组分成两部分，然后根据目标值与中间元素的比较结果来确定下一步查找的范围。如果目标值等于中间元素，则查找成功；如果目标值小于中间元素，则在左半部分继续查找；如果目标值大于中间元素，则在右半部分继续查找。重复这个过程，直到找到目标值或者查找范围为空。in 运算符的工作原理是通过遍历序列中的每个元素，并与要查找的元素进行比较，如果找到相等的元素，则返回 True，否则返回 False。所以本题错误。

5. 在以下 Python 代码中，最后一行代码执行时将报错，因为 y 所代表的数已经被删除。

```
x = [1, 2, 3, [4, 5, 6]]
y = x[3]

del x[3]
print(y)
```

【答案】错误×

【解析】本题考察 Python 列表删除的知识。del 只能删除对象的引用，而不能删除对象本身。所以[4,5,6]并没有被删除，最后一行代码正常执行。所以本题错误。

6. 贪心算法的解可能不是最优解。

【答案】正确√

【解析】本题考察贪心算法的知识。贪心算法是一种在每一步选择中都采取在当前状态下最好或最优的选择，从而希望导致结果是最好或最优的算法。然而，贪心算法并不总是能够找到全局最优解。在某些情况下，贪心算法可能会陷入局部最优解，而无法找到全局最优解。这是因为贪心算法在每一步都只考虑当前的情况，而没有考虑到未来的影响。因此，如果一个问题的最优解需要通过牺牲一些短期的利益来获得长期的利益，那么贪心算法就可能无法找到这个最优解。所以本题正确。

7. 一般说来，冒泡排序算法优于归并排序。

【答案】错误×

【解析】本题考察冒泡排序算法和归并排序知识。冒泡排序算法的优点是简单易懂、实现起来较为方便，但其时间复杂度较高，不适用于大规模数据的排序；而归并排序虽然实现起来较为复杂，但其时间复杂度较低，适合处理大规模数据的排序。因此，二者各有优势，不能简单地说哪个更优。具体使用哪种排序算法，需要根据实际问题的需求来决定。所以本题错误。

8. Python 的内置函数 sorted() 可以对支持 for-in 循环的 str、list、tuple、dict、set 排序，且是稳定排序。

【答案】正确√

【解析】本题考察 sorted( )函数相关知识和算法稳定概念，sorted()函数的实现基于 Timsort 算法，Timsort 算法在排序时，先将待排序的序列分割成若干个子序列，对每个子序列进行排序，然后将已排序的子序列合并成一个新的有序序列，直到整个序列排好序为止。是结合了合并排序和插入排序而得出的排序算法。如果一个排序确保不会改变比较结果相等的元素的相对顺序就称其为稳定的。故说法正确。

9. 下面的 Python 代码将输出 0-99（包含 0 和 99）之间的整数，顺序随机。

```
import random
print(sorted(range(100), key = lambda x:
random.random()))
```

【答案】正确√

【解析】本题属于考察 Python 中 sorted()函数，lamdba 函数使用的知识。sorted()函数的 key 参数可以接受一个函数作为值，可以根据自定义的函数逻辑对列表进行排序。random.random()是 Python 中的一个函数，它属于 random 模块。该函数用于生成一个 0 到 1 之间的随机浮点数，包含 0 但不包含 1。每次调用该函数都会返回一个新的随机数。所以本题正确。

10. 下面的 Python 代码执行后将输出 [0, 5, 1, 6, 2, 3, 4] 。

```
lst = list(range(7))
sorted(lst, key = lambda x: x%5)
print(lst)
```

【答案】错误×

【解析】本题属于考察 Python 中 sorted()函数的知识，sorted()是 Python 内置的一个函数，用于对可迭代对象进行排序操作。它返回一个新的已排序的列表，而不会修改原始的可迭代对象。故输出为[0,1,2,3,4,5,6]。所以本题错误。

三、编程题（每题 25 分，共 50 分）

1.因数分解

【问题描述】

每个正整数都可以分解成素数的乘积，例如：6=2×3、20=2^2 ×5。现在，给定一个正整数 N，请按要求输出它的因数分解式。

【输入描述】

输入第一行，包含一个正整数N。约定$2 \leq N \leq 10^{12}$

【输出描述】

输出一行，为 N 的因数分解式。要求按质因数由小到大排列，乘号用星号*表示，且左右各空一格。当且仅当一个素数出现多次时，将它们合并为指数形式，用上箭头^表示，且左右不空格。

【样例输入 1】

6

【样例输出 1】

2*3

【样例输入 2】

20

【样例输出 2】

2^2*5

【题目大意】

输出 N 的因数分解式，按质因数由小到大排列，乘号用星号*表示，且左右各空一格。当且仅当一个素数出现多次时，将它们合并为指数形式，用上箭头^表示，且左右不空格

【解题思路】

本题主要考察素因数相关算法的知识。

1. 输入一个数字，设置一个输出列表 factors。

2. 设定一个判断素数的函数 isPrime()。

3．如果是素数，将该数和数字 1 作为列表存入列表 factors。

4．否则，遍历从 2 到 input_number,如果 i 为 input_number 的因数，将该因数 i 和数字 0 存入列表 factors，使用唯一分解定理，找到素因数，将 0 增加 1。当因数不能整除时跳出 while 循环，判断该数是否素数，满足添加到列表中。

5．遍历 factors 列表，如果只出现 1 次，直接输出，否则换成指数形式输出

【参考程序】

```
input_number = int(input())
original_number = input_number
factors = []

def isPrime(num):
    for j in range(2, int(num ** 0.5)+1):
        if num % j == 0:
            return False
    return True

if isPrime(input_number):
    factors.append([input_number, 1])
else:
    for i in range(2, input_number + 1):
        input_number = int(input_number)
        if input_number % i == 0:
            factors.append([i, 0])
            while input_number % i == 0:
                factors[-1][1] += 1
                input_number //= i
            if isPrime(input_number):
                if input_number != 1:
                    factors.append([input_number, 1])
                break
for i in range(len(factors)):
    if i != 0:
        print("*", end=" ")
```

```
    if factors[i][1] == 1:
        print(factors[i][0], end=" ")
    else:
        print(f"{factors[i][0]}^{factors[i][1]}", end=" ")
print(" ")
```

2.巧夺大奖

【问题描述】

小明参加了一个巧夺大奖的游戏节目。主持人宣布了游戏规则：

1、游戏分为 n 个时间段，参加者每个时间段可以选择一个小游戏。

2、游戏中共有 n 个小游戏可供选择。

3、每个小游戏有规定的时限和奖励。对于第 i 个小游戏，参加者必须在第T_i个时间段结束前完成才能得到奖励R_i。

小明发现，这些小游戏都很简单，不管选择哪个小游戏，他都能在一个时间段内完成。关键问题在于，如何安排每个时间段分别选择哪个小游戏，才能使得总奖励最高？

【输入描述】

输入第一行，包含一个正整数 n。n 既是游戏时间段的个数，也是小游戏的个数。约定$1 \le n \le 500$。

输入第二行，包含 n 个正整数。第 i 个正整数为T_i，即第 i 个小游戏的完成期限。约定$1 \le T_i \le n$。

输入第三行，包含 n 个正整数。第 i 个正整数为R_i，即第 i 个小游戏的完成奖励。约定$1 \le R_i \le 1000$。

【输出描述】

输出一行，包含一个正整数 C，为最高可获得的奖励。

【样例输入 1】

```
7
4 2 4 3 1 4  6
70 60 50 40 30 20  10
```

【样例输出 1】

230

【样例解释 1】

7 个时间段可分别安排完成第 4、2、3、1、6、7、5 个小游戏，其中第 4、2、3、1、7 个小游戏在期限内完成。因此，可以获得总计$40+60+50+70+10=230$的奖励。

【题目大意】

在 n 个时间段，有 n 个游戏可以选择，每个游戏有规定的时限和奖励，小游戏都很简单，不管选择哪个小游戏，都能在一个时间段内完成。如何安排每个时间段分别选择哪个小游戏，才能总奖励最高。例如，第 1 个小游戏，完成期限是 4，奖励是 7，即第 1 个小游戏要在第 4 个游戏结束前完成才可以得到奖励 7。

【解题思路】

本题考察贪心算法问题。

1. 输入总分段数 total_segements、结束时间 s_limits 和奖励值 s_rewards，将结束时间和奖励值遍历存入到 limits 和 rewards。然后根据这些信息构建一个 game 列表，每个列表项为[结束时间，奖励值，是否可选，是否选过]。

2. 定义函数 check_chooseable(),check_chooseable()函数用于检查当前状态下是否可以选择某个游戏。

3．定义函数 get_current_max()，get_current_max()函数用于获取当前状态下可以选择的最大奖励。

4．从 total_segements 开始循环遍历所有可能的分段数，计算最大奖励。

【参考程序】

```
total_segements = int(input())
limits = []
rewards = []
s_limits = input()
s_limits = s_limits.split(" ")
s_rewards = input()
s_rewards = s_rewards.split(" ")
for item in s_limits:
    limits.append(int(item))
for item in s_rewards:
```

```
    rewards.append(int(item))
games = []
for i in range(len(s_limits)):
    games.append([limits[i], rewards[i], False, False])


def check_chooseable(states, current):
    for item in states:
        if item[0] >= current:
            item[2] = True


def get_current_max(states, current):
    maximum = 0
    chosen = None
    for i in range(len(states)):
        if states[i][2] and not states[i][3]:
            if maximum < states[i][1]:
                maximum = max(states[i][1], maximum)
                chosen = i
    if chosen is not None:
        states[chosen][3] = True
    return maximum


total = 0
for t in range(total_segements, 0, -1):
    check_chooseable(games, t)
    total += get_current_max(games, t)
print(total)
```