

2023 年 9 月 GESPC++ 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 C++ 五级认证真题解析。

一、单选题（每题 2 分，共 30 分）

题号	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
答案	A	D	C	C	A	B	A	C	C	D	B	C	B	A	C

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

【考纲知识点】 计算机基础知识

【解析】 本题属于考察计算机基础知识知识。手写板是输入信息的设备，选 A。

2、如果 a 和 b 均为 `int` 类型的变量，且 b 的值不为 0 ，那么下列能正确判断“ a 是 b 的 3 倍”的表达式是（ ）。

- A. $(a \gg 3 == b)$
- B. $(a - b) \% 3 == 0$
- C. $(a / b == 3)$
- D. $(a == 3 * b)$

【答案】D

【考纲知识点】 运算表达式和位运算

【解析】 本题属于考察运算表达式和位运算知识。 b 不等于 0 ， a 是 b 的 3 倍。
A 选项中， a 右移 3 位，相当于除以 8 ；B 是取余运算；如果 $a=7, b=2, a/b$ 的结果也等于 3 ，因为是整型，C 选项也不正确；选 D。

3、如果变量 a 和 b 分别为 `double` 类型和 `int` 类型，则表达式

$(a = 6, b = 3 * (7 + 8) / 2, b += a)$ 的计算结果为（ ）。

- A. 6
- B. 21
- C. 28
- D. 不确定

【答案】C

【考纲知识点】 数据类型和运算表达式

【解析】 本题属于考察数据类型、逗号表达式、运算表达式知识。逗号表达式从左到右依次计算每一个表达式，整个逗号表达式的结果是最后一个表达式的计算结果， $a=6, b=3*15/2=22, b=6+22=28$ ，选 C。

4、有关下面C++代码说法错误的是（ ）。



```
1 // sumA()和sumB()用于求从1到N之和
2 #include <iostream>
3 using namespace std;
4 int sumA(int n) {
5     int sum = 0;
6     for (int i = 1; i < n + 1; i++)
7         sum += i;
8     return sum;
9 }
10 int sumB(int n) {
11     if (n == 1)
12         return 1;
13     else
14         return n + sumB(n - 1);
15 }
16 int main() {
17     int n = 0;
18     cin >> n;
19     cout << sumA(n) << " " << sumB(n) << endl;
20     return 0;
21 }
```

- A. sumA() 用循环方式求从 1 到 N 之和， sumB() 用递归方式求从 1 到 N 之和。
- B. 默认情况下，如果输入正整数 1000 ， 能实现求从 1 到 1000 之和。
- C. 默认情况下，如果输入正整数 100000 ， 能实现求从 1 到 100000 之和。
- D. 一般说来， sumA() 的效率高于 sumB() 。

【答案】 C

【考纲知识点】 函数和递归算法

【解析】本题属于考察递归和函数知识。sumA 用循环求的是 1 到 n 的总和；sumB 用递归的方式求 1 到 n 的总和。1 加到 100000 的和大于 int 类型最大值，2147483647，会溢出，选 C。

5、下面C++代码以递归方式实现字符串反序，横线处应填上代码是（ ）。



```
1 //字符串反序
2 #include <iostream>
3 #include <string>
4 using namespace std;
5 string sReverse(string sIn) {
6     if (sIn.length() <= 1) {
7         return sIn;
8     } else {
9         return _____ // 此处填写代码
10    }
11 }
12 int main() {
13     string sIn;
14     cin >> sIn;
15     cout << sReverse(sIn) << endl;
16     return 0;
17 }
```

- A. `sIn[sIn.length() - 1] + sReverse(sIn.substr(0, sIn.length() - 1));`
- B. `sIn[0] + sReverse(sIn.substr(1, sIn.length() - 1));`
- C. `sReverse(sIn.substr(0, sIn.length() - 1)) + sIn[sIn.length() - 1];`
- D. `sReverse(sIn.substr(1, sIn.length() - 1)) + sIn[sIn.length() - 1];`

【答案】 A

【考纲知识点】 函数和递归算法

【解析】本题属于考察递归和函数知识。要倒序输出字符串，先输出字符串中的最后一个字符，最后一个字符下标是长度-1，然后翻转除了最后一个字符的字符串，这个字符串是从0开始，选 `sIn.length()-1` 个字符，选 A。

6、印度古老的汉诺塔传说：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 C++ 代码以递归方式实现汉诺塔，横线处应填入代码是 ()。



```
1  #include <iostream>
2  using namespace std;
3  // 递归实现汉诺塔，将N个圆盘从A通过B移动C
4  // 圆盘从底到顶，半径必须从大到小
5  void Hanoi(string A, string B, string C, int N) {
6      if (N == 1) {
7          cout << A << " -> " << C << endl;
8      } else {
9          Hanoi(A, C, B, N - 1);
10         cout << A << " -> " << C << endl;
11         _____; // 此处填写代码
12     }
13 }
14 int main() {
15     Hanoi("甲", "乙", "丙", 3);
16     return 0;
17 }
```

- 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

【考纲知识点】 函数和递归算法

【解析】 本题属于考察递归和函数知识。汉诺塔比较经典的递归问题，第 11 行代码意味这剩下的 N-1 快圆盘，现在 B 柱子上，借助 A 柱，挪到 C 柱上，选 B。

7、根据下面C++代码的注释，两个横线处应分别填入（ ）。



```
1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4  using namespace std;
5
6  bool isOdd(int N) {
7      return N % 2 == 1;
8  }
9  bool compare(int a, int b) {
10     if (a % 2 == 0 && b % 2 == 1)
11         return true;
12     return false;
13 }
14 int main() {
15     vector<int> lstA; // lstA是一个整型向量
16     for (int i = 1; i < 100; i++)
17         lstA.push_back(i);
18     // 对lstA成员按比较函数执行结果排序
19     sort(lstA.begin(), lstA.end(), _____); // 此处填写代码1
20
21     vector<int> lstB;
22     for (int i = 0; i < lstA.size(); i++) // lstB成员全为奇数
23         if (_____) // 此处填写代码2
24             lstB.push_back(lstA[i]);
25
26     cout << "lstA: ";
27     for (int i = 0; i < lstA.size(); i++)
28         cout << lstA[i] << " ";
29     cout << endl;
30
31     cout << "lstB: ";
32     for (int i = 0; i < lstB.size(); i++)
33         cout << lstB[i] << " ";
34     cout << endl;
35     return 0;
36 }
```

- A. compare 和 isOdd(lstA[i])
- B. compare(x1,y1) 和 isOdd
- C. compare 和 isOdd
- D. compare(x1,y1) 和 isOdd(lstA[i])

【答案】A

【考纲知识点】 函数的调用、值传递

【解析】本题考察的是 `vector` 和 `sort` 函数的使用。本题中首先创建了一个 `vector lstA` 存储整数 1~99，然后调用 `sort` 函数对 `lstA` 进行排序，`sort` 函数排序需要传递三个参数，前两个参数决定了排序范围的起始位置和结束位置，第三个参数是排序规则函数，排序规则函数需要两个参数和一个 `bool` 类型的返回值，所以代码 1 处传递排序规则函数 `compare`，只需要写函数名即可，不需要写成调用的形式，排除选项 B 和 D。

接下来的代码是将 `lstA` 中的奇数存储到另一个 `vector lstB` 中，判断奇数可以使用自定义函数 `isOdd`，代码 2 处是调用 `isOdd` 函数判断当前的 `lstA[i]` 是否为奇数，因此需要使用调用函数语法 `isOdd(lstA[i])`，正确答案为 A 选项。

8、有关下面代码正确的是（ ）。

```
1 // 在C++语言中，可以通过函数指针的形式，将一个函数作为另一个函数的参数。
2 // 具体来说: bool checkNum(bool (*Fx)(int), int N); 声明了一个函数，
3 // 其第一个参数是函数指针类型，指向一个接收一个int参数且返回值为bool的函数。
4 #include <iostream>
5 using namespace std;
6
7 bool isEven(int N) {
8     return N % 2 == 0;
9 }
10 bool checkNum(bool (*Fx)(int), int N) {
11     return Fx(N);
12 }
13 int main() {
14     cout << checkNum(isEven, 10) << endl;
15     return 0;
16 }
```

- A. `checkNum()` 函数定义错误。
- B. 将 `isEven` 作为 `checkNum()` 参数将导致错误。
- C. 执行后将输出 1 。
- D. 运行时触发异常。

【答案】C

【考纲知识点】 函数的定义、调用、值传递

【解析】本题考察函数指针的知识。isEven 是一个自定义函数，用于判断为偶数。checkNum 也是一个自定义函数，该函数的第一个参数是一个函数指针类型，需要传递一个函数，所传递的这个函数要求返回值为 bool，并且有一个 int 类型参数，isEven 函数复合该类型的要求，所以调用 checkNum 函数时，第一个参数传递函数 isEven，第二个参数传递整数 10 没有错误，执行 checkNum 函数内部代码 Fx(N)，也就是调用函数 isEven(10)，整数 10 是偶数，因此最后调用的结果是 1，因此正确答案选 C。

9、有关下面C++代码正确的是（ ）。

```
1  #include <iostream>
2  using namespace std;
3
4  bool isOdd(int N) {
5      |   return N % 2 == 1;
6  }
7  int Square(int N) {
8      |   return N * N;
9  }
10 bool checkNum(bool (*Fx)(int), int x) {
11     |   return Fx(x);
12 }
13 int main() {
14     |   cout << checkNum(isOdd, 10) << endl;    // 输出行A
15     |   cout << checkNum(Square, 10) << endl;  // 输出行B
16     |   return 0;
17 }
```

- A. checkNum() 函数定义错误。
- B. 输出行 A 的语句将导致编译错误。
- C. 输出行 B 的语句将导致编译错误。
- D. 该代码没有编译错误。

【答案】c

【考纲知识点】 函数指针

【解析】本题考察函数指针的知识。checkNum 函数的第一个参数需要传递一个返回值为 bool、参数为 int 类型的函数，isEven 函数复合该类型的要求，所以调用 checkNum 函数时，第一个参数传递函数 isEven，不会导致编译错误，Square 函数的返回值是 int 类型，不符合 checkNum 函数的第一个参数的类型要求，因此调用时传递 Square 函数会导致编译错误，所以答案为 C 选项。

10、下面代码执行后的输出是（ ）。

```
1  #include <iostream>
2  using namespace std;
3
4  int jumpFloor(int N) {
5      cout << N << "#";
6      if (N == 1 || N == 2) {
7          return N;
8      } else {
9          return jumpFloor(N - 1) + jumpFloor(N - 2);
10     }
11 }
12 int main() {
13     cout << jumpFloor(4) << endl;
14     return 0;
15 }
```

- 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

【考纲知识点】 递归和函数

【解析】本题属于考察递归和函数知识。第 13 行代码调用 jumFloor(4)，首先输出 4#，然后返回 jumFloor(3)+ jumFloor(2)，jumFloor(3)输出 3#再继续递归调用 jumFloor(2)+ jumFloor(1)依次输出 2#1#并返回 3。

jumFloor(2)输出 2#返回 2，所以第 13 行代码调用之后的返回结果是 3+2=5，最后的输出结果就是 4#3#2#1#2#5，所以正确答案选 D。

11、下面代码中的 `isPrimeA()` 和 `isPrimeB()` 都用于判断参数 `N` 是否素数，有关其时间复杂度的正确说法是（ ）。

```
1  #include <iostream>
2  #include <cmath>
3  using namespace std;
4
5  bool isPrimeA(int N) {
6      if (N < 2)
7          return false;
8      for (int i = 2; i < N; i++)
9          if (N % i == 0)
10             return false;
11     return true;
12 }
13 bool isPrimeB(int N) {
14     if (N < 2)
15         return false;
16     int endNum = int(sqrt(N));
17     for (int i = 2; i <= endNum; i++)
18         if (N % i == 0)
19             return false;
20     return true;
21 }
22 int main() {
23     cout << boolalpha;
24     cout << isPrimeA(13) << " " << isPrimeB(13) << endl;
25     return 0;
26 }
```

- A. `isPrimeA()` 的最坏时间复杂度是 $O(N)$ ，`isPrimeB()` 的最坏时间复杂度是 $O(\log N)$ ，`isPrimeB()` 优于 `isPrimeA()`。
- B. `isPrimeA()` 的最坏时间复杂度是 $O(N)$ ，`isPrimeB()` 的最坏时间复杂度是 $O(N^{1/2})$ ，`isPrimeB()` 优于 `isPrimeA()`。
- C. `isPrimeA()` 的最坏时间复杂度是 $O(N^{1/2})$ ，`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 $O(N)$ ，`isPrimeA()` 优于 `isPrimeB()`。

【答案】 B

【考纲知识点】 时间复杂度

【解析】 本题属于考察时间复杂度的相关知识。**isPrimeA** 的最坏时间复杂度是 $O(N)$ ，**isPrimeB** 的最坏时间复杂度是 $O(1)$ ， $O(1)$ 优于 $O(N)$ ，正确答案 **B** 选项。

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



```
1  #include <iostream>
2  using namespace std;
3
4  void mergeSort(int * listData, int start, int end);
5  void merge(int * listData, int start, int middle, int end);
6
7  void mergeSort(int * listData, int start, int end) {
8      if (start >= end)
9          return;
10     int middle = (start + end) / 2;
11     mergeSort(listData, start, middle);
12     mergeSort(listData, middle + 1, end);
13     merge(listData, start, middle, end);
14 }
15 void merge(int * listData, int start, int middle, int end) {
16     int leftSize = middle - start + 1;
17     int rightSize = end - middle;
18
19     int * left = new int[leftSize];
20     int * right = new int[rightSize];
21     for (int i = 0; i < leftSize; i++)
22         left[i] = listData[start + i];
23     for (int j = 0; j < rightSize; j++)
24         right[j] = listData[middle + 1 + j];
25
26     int i = 0, j = 0, k = start;
27     while (i < leftSize && j < rightSize) {
28         if (left[i] <= right[j]) {
29             listData[k] = left[i];
30             i++;
31         } else {
32             listData[k] = right[j];
33             j++;
34         }
35         k++;
36     }
37
38     while (i < leftSize) {
39         listData[k] = left[i];
40         i++;
41         k++;
42     }
43     while (j < rightSize) {
44         listData[k] = right[j];
45         j++;
46         k++;
47     }
48     delete[] left;
49     delete[] right;
50 }
51 int main() {
52     int lstA[] = {1, 3, 2, 7, 11, 5, 3};
53     int size = sizeof(lstA) / sizeof(lstA[0]);
54
55     mergeSort(lstA, 0, size - 1); // 对lstA执行归并排序
56
57     for (int i = 0; i < size; i++)
58         cout << lstA[i] << " ";
59     cout << endl;
60     return 0;
61 }
```

- A. 0
- B. 1
- C. 6
- D. 7

【答案】C

【考纲知识点】 分治算法

【解析】 本题考察归并排序的相关知识。listA 的长度为 7，根据递归返回的条件 $start \geq end$ ，merge 函数会被调用 6 次，正确答案 C 选项。

13、在上题的归并排序算法中，mergeSort(listData, start, middle); 和 mergeSort(listData, middle + 1, end); 涉及到的算法为（ ）。

- A. 搜索算法
- B. 分治算法
- C. 贪心算法
- D. 递推算法

【答案】B

【考纲知识点】

【解析】 本题属于考察归并排序的相关知识。归并排序采用的是分治算法思想，正确答案 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

【考纲知识点】 归并排序算法

【解析】 本题属于考察归并排序的相关知识。归并排序基本思想就是将数组分成两个子数组，分别排序后再合并，正确答案 A 选项。

15、有关下面代码的说法正确的是（ ）。

```

1  #include <iostream>
2
3  class Node {
4  public:
5      int Value;
6      Node * Next;
7
8      Node(int Val, Node * Nxt = nullptr) {
9          Value = Val;
10         Next = Nxt;
11     }
12 };
13
14 int main() {
15     Node * firstNode = new Node(10);
16     firstNode->Next = new Node(100);
17     firstNode->Next->Next = new Node(111, firstNode);
18     return 0;
19 }

```

- 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

【考纲知识点】 循环链表

【解析】 本题考察链表的相关知识。第 15 行代码~第 17 行代码创建了三个链表的节点，第一个节点的 next 指向第二个节点，第二个节点的 next 指向第三个节点，第三个节点的 next 又指向了第一个节点，形成了一个环，所以是循环链表，正确答案 C 选项。

二、判断题（每题 2 分，共 20 分）

题号	1	2	3	4	5	6	7	8	9	10
答案	√	×	×	×	√	√	×	√	√	√

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

【考纲知识点】 计算机网络

【解析】 本题是计算机网络的知识，传输层是这 2 个协议。

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

【考纲知识点】 流程图

【解析】 本题考察流程图，流程图中没有三角框。

3、找出自然数 N 以内的所有质数，常用算法有埃氏筛法和线性筛法，其中埃氏筛法效率更高。

【答案】 错误

【考纲知识点】 线性筛法和埃氏筛法

【解析】 本题考察筛选素数的算法，线性筛法是在埃氏筛法基础的改进，效率更高。

4、在 C++ 中，可以使用二分法查找链表中的元素。

【答案】 错误

【考纲知识点】 二分法和链表

【解析】 本题考察二分法和链表的知识点，使用二分法查找元素，元素必须是顺序存储的，链表不是顺序存储数据，因此不能使用二分法。

5、在 C++ 中，通过恰当的实现，可以将链表首尾相接，形成循环链表。

【答案】 正确

【考纲知识点】 循环链表

【解析】 本题考察循环链表的知识点，链表的最后一个节点的 `next` 指针指向头结点就能形成循环链表。

6、贪心算法的解可能不是最优解。

【答案】正确

【考纲知识点】 贪心算法

【解析】 本题考察贪心算法的知识点，贪心算法找到的不一定是最优解。

7、一般说来，冒泡排序算法优于归并排序。

【答案】错误

【考纲知识点】 冒泡排序

【解析】 本题考察排序算法性能，冒泡时间复杂度为 $O(N^2)$ ，归并排序的时间复杂度为 $O(N\log N)$ ，因此归并优于冒泡，说法错误。

8、C++语言中的 `qsort` 库函数是不稳定排序。

【答案】正确

【考纲知识点】 `qsort` 函数

【解析】 本题考察 `qsort` 函数，`qsort` 函数内部使用的是不稳定的排序算法。

9、质数的判定和筛法的目的并不相同，质数判定旨在判断特定的正整数是否为质数，而质数筛法意在筛选出范围内的所有质数。

【答案】正确

【考纲知识点】 质数的判定和筛选

【解析】 本题考察质数的判定和筛选，说法正确。

10、下面的 C++ 代码执行后将输出 0 5 1 6 2 3 4。



```
1  #include <iostream>
2  #include <algorithm>
3  using namespace std;
4
5  bool compareModulo5(int a, int b) {
6  |   return a % 5 < b % 5;
7  }
8  int main() {
9  |   int lst[7];
10 |   for (int i = 0; i < 7; i++)
11 |       lst[i] = i;
12
13 |   // 对序列所有元素按compareModulo5结果排序
14 |   sort(lst, lst + 7, compareModulo5);
15
16 |   for (int i = 0; i < 7; i++)
17 |       cout << lst[i] << " ";
18 |   cout << endl;
19 |   return 0;
20 }
```

【答案】正确

【考纲知识点】 sort 函数

【解析】本题考察 sort 函数的排序规则，排序规则函数 compareModule5 确定的排序规则是根据除 5 的余数进行升序排序，所以 main 函数中对 0~6 范围的整数排序之后的结果就是 0516234。

三、编程题（每题 25 分，共 50 分）

题号	1	2
答案		

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： $6=2\times 3$ 、 $20=2^2\times 5$ 。

现在，给定一个正整数 N ，请按要求输出它的因数分解式 $2\leq N\leq 10^{12}$ 。

输入描述

输入第一行，包含一个正整数 N 。约定

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

```
1 | 6
```

样例输出 1

```
1 | 2 * 3
```

样例输入 2

```
1 | 20
```

样例输出 2

```
1 | 2^2 * 5
```

样例输入 3

```
1 | 23
```

样例输出 3

```
1 | 23
```

【题目大意】 输入一个正整数 N ，按格式输出它的因数分解式。

【考纲知识点】 初等数论，多重循环，算术运算

【解题思路】

每个正整数 N 的质因数分解形式是唯一的。可以设计一个简单的算法，在 $2 \sim N$ 范围内按从小到大的顺序枚举每一个整数，如果该整数能整除 N ，则把该整数就是 N 的一个质因数，将它从 N 中分解出去，循环执行直到 N 不能被分解为止。再分解过程中按题目要求输出因数分解式。

【参考程序】



```
1 #include <iostream>
2 using namespace std;
3 int main() {
4     long long N = 0;
5     cin >> N;
6     bool first = true;
7     for (long long p = 2; p * p <= N; p++) {
8         if (N % p != 0)
9             continue;
10        int cnt = 0;
11        while (N % p == 0) {
12            cnt++;
13            N /= p;
14        }
15        if (first) {
16            first = false;
17        } else {
18            cout << " * ";
19        }
20        cout << p;
21        if (cnt > 1)
22            cout << "^" << cnt;
23    }
24    if (N > 1) {
25        if (first) {
26            first = false;
27        } else {
28            cout << " * ";
29        }
30        cout << N;
31    }
32    cout << endl;
33    return 0;
34 }
```

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 n 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 $1 \leq n \leq 500$ 。

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 $1 \leq T_i \leq n$ 。

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 $1 \leq R_i \leq 1000$ 。

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

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

样例输出 1

```
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 个时间段内完成 n 个小游戏，每个小游戏完成的时间和获得奖励不同，如何选择小游戏使得最后获得奖励最高。

需要注意这句话的理解“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”，也就是在第 $1 \sim T_i$ 个时间段范围之内，其中任意一个时间段都可以完成第 i 个游戏。

【考纲知识点】 贪心算法、数组、`sort` 函数。

【解题思路】

本题采用贪心策略，想要获得的最高奖励，优先完成获得奖励多的游戏，同时考虑在完成第 i 个游戏的时候，第 $1 \sim T_i$ 个时间段是否被占用，如果都被占用，那么该游戏就不能被完成。解题步骤如下：

1) 首先创建结构体 `game` 用于保存每个游戏的信息，包括游戏时间期限 T 和对应的奖励 R ，并创建 `games[505]` 用于保存 n 个游戏的信息

- 2) 按题目要求输入数据，并保存在 `games` 数组中
- 3) 根据游戏的奖励，对数组 `games` 进行降序排序，
- 4) 遍历排序后的数组 `games`，依次判断第 i 个游戏是否能完成，如果能完成就累加当前游戏的奖励 `games[i].R`
- 5) 判断游戏是否能完成可以使用一个数组进行标记，标记第 $1n$ 个时间段是否被占用，如果第 i 个游戏的可完成时间段为第 $1Ti$ ，如果该范围都被占用，则第 i 个游戏无法完成。

【参考程序】

```
1 #include <iostream>
2 #include <algorithm>
3 using namespace std;
4 int n = 0;
5 struct game_t {
6     int T, R;
7 } games[500];
8 bool game_cmp(game_t x, game_t y){
9     return x.R > y.R;
10 }
11 bool arrange[500];
12 int main() {
13     cin >> n;
14     for (int i = 0; i < n; i++)
15         arrange[i] = false;
16     for (int i = 0; i < n; i++)
17         cin >> games[i].T;
18     for (int i = 0; i < n; i++)
19         cin >> games[i].R;
20     sort(games, games + n, game_cmp);
21     int sum = 0;
22     for (int i = 0; i < n; i++) {
23         for (int t = games[i].T - 1; t >= 0; t--)
24             if (!arrange[t]) {
25                 arrange[t] = true;
26                 sum += games[i].R;
27                 break;
28             }
29     }
30     cout << sum << endl;
31     return 0;
32 }
```