### 一 活动选择问题：

#### 1伪代码描述：

递归贪心算法：

// 设所有活动按结束时间递增排列，该序列包含活动

// 每个活动开始时间依次保存在s中，结束时间依次保存在f中

// 找到活动序列中开始的所有活动的一个最大兼容活动子集

REC-ACTIVITY-SELECTOR(s, f, k, n)

m = k + 1

while m <= n and s[m] < f[k] do

m = m + 1

if m <= n then

return { ak } ∪ REC-ACTIVITY-SELECTOR(s, f, m, n)

return { ak }

迭代贪心算法：

// 假设至少有1个活动，所有活动按结束时间递增排列

// 每个活动开始时间依次保存在s中，结束时间依次保存在f中

// 返回所有活动的一个最大兼容活动子集

ITE-ACTIVITY-SELECTOR(s, f)

A = { a1 }

k = 1

for m = 2 to |s| do

if s[m] >= f[k] then

A = A ∪ { am }

k = m

return A

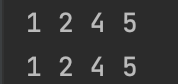
#### 2具体输入格式：

输入只需要修改活动个数n，和活动开始时间和活动结束时间两个数组中元素的值。



#### 3具体输出格式：

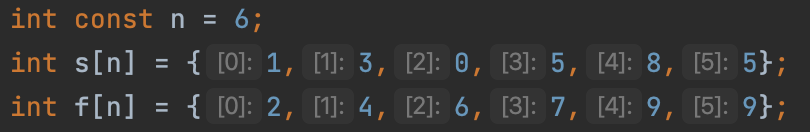
依次输出迭代贪心算法和递归贪心算法找到的最大兼容活动子集。



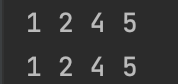
#### 4测试用例：

1.

输入：

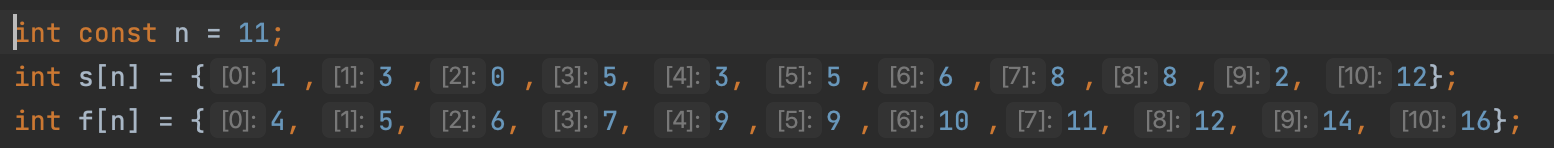


输出：

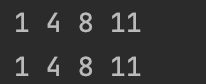


2

输入：

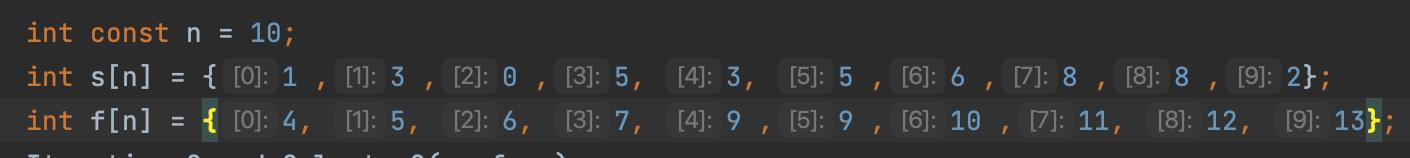


输出：

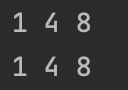


3

输入：



输出：



### 二 子集合问题：

#### 1伪代码描述：

// A为给定正整数集合中元素升序排序所得数组

// 输出包含数组B中所有元素，其他元素在A[i .. ]中，元素之和为d的子集

// |A|表示数组A中包含有效元素的个数

//++表示两个数组的连接

//A[i .. ]表示A中下标i开始的子数组

// SUM(…)表示数组中元素的和

SUBSET-SUM(A, d, B, i)

s = SUM(B)

if s == d then

print B

return

if i >= |A| or s + A[i] > d or s + SUM(A[i .. ]) < d then

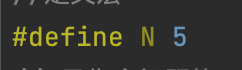
return

SUBSET-SUM(A, d, B++[A[i]], i+1)

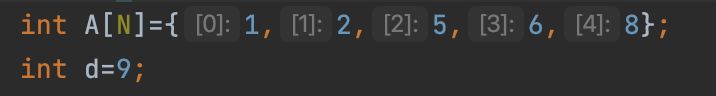
SUBSET-SUM(A, d, B, i+1)

#### 2具体输入格式：

先修改宏变量N为集合中的元素个数

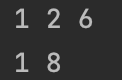


再修改集合中的所有元素和正整数d



#### 3具体输出格式：

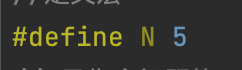
给出A的子集中元素的和等于某个正整数所有子集。

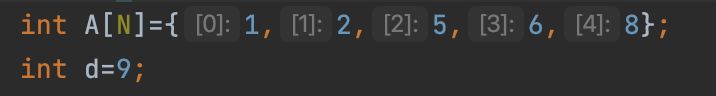


#### 4测试用例：

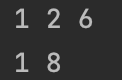
1.

输入：





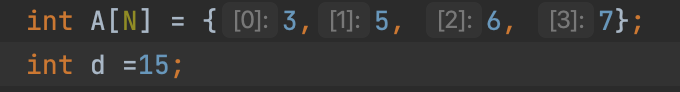
输出：



2

输入：



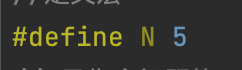


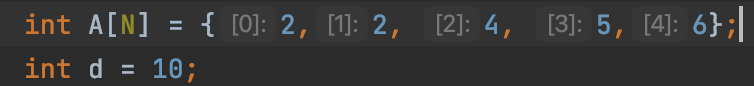
输出：



3

输入：





输出：

