程式設計（112-1）
作業三

作業設計：孔令傑
國立臺灣大學資訊管理學系

繳交作業時，請至 PDOGS（`http://pdogs.ntu.im/`）為第一、二、三、四題各上傳一份 C++ 原始碼（以複製貼上原始碼的方式上傳）。每位學生都要上傳自己寫的解答。不接受紙本繳交；不接受遲交。這份作業的截止時間是 ***9 月 26 日早上八點***。為這份作業設計測試資料並且提供解答的助教是王裕勳。

在你開始前，請閱讀課本的第 5.20–5.22（關於遞迴）和第 19 章（關於搜尋和排序）[1]。

本次作業滿分為 110 分，得幾分就算幾分。若整學期有 n 份作業，則學期的作業總成績即為 n 份作業的總分除以 n（不論超過 100 與否）。

第一題

（20 分）在 knapsack 這個問題中[2]，我們要決定該如何選擇物品，並將選定的物品放入耐重度有限的背包之中。題目會給定 n 件物品，每一件物品都有自己的價值 v_i 以及重量 w_i，並且這些物品都不可被分割。已知背包的耐重度上限為 B，我們希望可以選擇若干件物品，在不超過背包耐重度的前提之下，最大化背包內物品的總價值。換句話說，令 $x_i = 1$ 表示要選物品 i，$x_i = 0$ 則否，我們希望求解以下最佳化問題：

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} v_i x_i \\ \text{s.t.} \quad & \sum_{i=1}^{n} w_i x_i \le B \\ & x_i \in \{0,1\} \quad \forall i = 1, ..., n。 \end{aligned}$$

如同上課所說，要找出 knapsack 的最佳解是一件困難的事（也就是 knapsack 是所謂的 NP-hard problem），因此本題會給定好一組解，你的程式要判定這個解是否為可行解（feasible solution），也就是檢查將這些物品全部放進背包時，總重量是否會超過耐重度上限。如果為可行解，則進一步計算所有被選取物品的總重量與總價值。

在本題中，你的夥伴又已經把 main function 寫好了，要求你按照他的設計寫一個函數；他會把讀入的資料存入對應的變數或陣列，接著把這些變數和陣列傳進你寫的函數，讓你做計算並且回傳是否超過耐重度上限，沒有的話請計算這個可行解的總重量與總價值。

具體來說，首先我們定義 `MAX_ITEM_CNT` 這個 constant 來代表物品數量的最大可能的值，接著定義這個函數的 prototype 為

```
void knapsack(int itemCnt, int capacity, const int weight[], const int
    value[], const bool bring[], int ret[]);
```

[1] 課本是 Deitel and Deitel 著的 *C++ How to Program: Late Objects Version* 第七版。
[2] 請參考「Week 01 Variables and Arrays」的 In-class Practices Problem 10。

其中 itemCnt 代表物品的總數量，capacity 是背包耐重度上限，weight 是一個長度為 MAX_ITEM_CNT 但應該要裝著 itemCnt 個元素的一維陣列，裡面每個元素就是第一號、第二號直到最後一號物品的重量，value 是一個長度為 MAX_ITEM_CNT 但應該要裝著 itemCnt 個元素的一維陣列，裡面每個元素就是第一號、第二號直到最後一號物品的價值，bring 是一個長度為 MAX_ITEM_CNT 但應該要裝著 itemCnt 個元素的一維陣列，裡面每個元素依序代表著是否選取第一號、第二號直到最後一號物品。最後 ret 是一個長度為 2 的一維陣列，在傳入時裡面存什麼都無所謂，但函數執行完畢時裡面需要存著函數想要回傳的兩個值（因應 C++ 的函數只能有一個回傳值的設計，這是一種回傳多個值的方法），如果此解超過耐重度上限，也就是不為可行解，ret[0] 就存 −1，而 ret[1] 存什麼都無所謂；反之，若為可行解，請於 ret[0] 存選取物品的總重量，再在 ret[1] 存選取物品的總價值。

你的夥伴寫的 main function 如下（如附件 PD112-1_hw03_main01.cpp）：

```cpp
#include <iostream>
using namespace std;
const int MAX_ITEM_CNT = 100;

// This is the prototype of the function that you should complete
void knapsack(int itemCnt, int capacity, const int weight[], const int
    value[], const bool bring[], int ret[]);

int main()
{
    int itemCnt = 0, capacity = 0;
    cin >> itemCnt >> capacity;

    int weight[MAX_ITEM_CNT] = {0};
    int value[MAX_ITEM_CNT] = {0};
    bool bring[MAX_ITEM_CNT] = {0};
    int ret[2] = {0};

    for(int i = 0; i < itemCnt; i++)
        cin >> weight[i];
    for(int i = 0; i < itemCnt; i++)
        cin >> value[i];
    for(int i = 0; i < itemCnt; i++)
        cin >> bring[i];

    knapsack(itemCnt, capacity, weight, value, bring, ret);

    if(ret[0] == -1)
        cout << -1;
    else
        cout << ret[0] << "," << ret[1];
```

```
    return 0;
}

// PDOGS will copy and paste your codes here before compilation
```

特別注意：在這題之中，助教已經在 PDOGS 上設定好上面的「你的夥伴」寫的程式了。你需要完成一個完整的 `knapsack` 函數，自己測試的時候當然需要結合上面的 main function，但在繳交到 PDOGS 時請只上傳這個 `knapsack` 函數，PDOGS 會自動把你上傳的函數跟已經在 PDOGS 上的程式拼起來去編譯。換言之，在本題你被迫必須要實作本題指定的函數；如果你上傳了任何帶有你寫的 main function 的程式，你會無法得到分數的！

輸入輸出格式

系統會提供一共 10 組測試資料，每組測試資料裝在一個檔案裡。在每個檔案中會有四列，第一列裝著兩個正整數，依序是 n、B，代表物品數量以及耐重度上限；第二列存了 n 個非負整數，依序是 w_1、w_2 直到 w_n，分別代表每一個物品的重量；第三列存了 n 個非負整數，依序是 v_1、v_2 直到 v_n，分別代表每一個物品的價值；第四列存了 n 個非負整數，依序是 x_1、x_2 直到 x_n，若 $x_i = 1$，代表這個解有將編號第 i 號物品放進背包中；反之若 $x_i = 0$，代表這個解沒有選取編號第 i 號物品。已知 $1 \le n \le 100$、$1 \le B \le 10000$、$1 \le w_i \le 100$、$1 \le v_i \le 100$、$x_i \in \{0, 1\}$。每一列的任兩個相鄰的整數間以一個空白字元隔開。

請依題目要求與指示，重新撰寫程式碼，並輸出這個解是否超過耐重度上限，若為可行解（沒有超過耐重度上限），請依序輸出選取物品的總重量與總價值，兩個數字之間以一個逗號隔開；若超過背包耐重度上限，請直接輸出 -1。舉例來說，如果輸入是

```
4 9
2 3 4 3
2 4 5 3
1 0 1 1
```

則輸出應該是

```
9,10
```

如果輸入是

```
4 9
3 4 2 3
5 4 3 2
1 1 1 0
```

則輸出應該是

```
9,12
```

如果輸入是

```
4 9
2 3 4 3
2 4 5 3
1 1 1 1
```

則輸出應該是

```
-1
```

你上傳的原始碼裡應該包含什麼

你的 .cpp 原始碼檔案裡面應該包含讀取測試資料、做運算，以及輸出答案的 C++ 程式碼。當然，你應該寫適當的註解。針對這個題目，你可以使用任何方法。

評分原則

這一題的所有分數都根據程式運算的正確性給分。PDOGS 會編譯並執行你的程式、輸入測試資料，並檢查輸出的答案的正確性。一筆測試資料佔 2 分。

第二題

（20 分）承上題，本題一樣會給定 n 件物品、耐重度上限 B，以及每一件物品的重量 w_i 與價值 v_i，不同的是，本題會提供 m 組解，每一組解都會選取若干件物品放入背包中。請寫一個程式判斷總共有多少組解為可行解（不超過背包耐重度上限），同時在這些可行解中，比較每一個解的選取物品總價值，並輸出最大的總價值為何。

輸入輸出格式

系統會提供一共 10 組測試資料，每組測試資料裝在一個檔案裡。在每個檔案中會有 $m+3$ 列，第一列裝著三個正整數，依序是 n、B、m，代表物品數量、耐重度上限以及解數量；第二列存了 n 個非負整數，依序是 w_1、w_2 直到 w_n，分別代表每一個物品的重量；第三列存了 n 個非負整數，依序是 v_1、v_2 直到 v_n，分別代表每一個物品的價值；第四列到第 $m+3$ 列中的第 j 列代表第 j 個解的相關資訊，在一列中會先有一個整數 q_j，代表這個解選了幾個物品，接著會有 q_j 個不重複但未必有排序的整數，代表被選中的物品的編號（第一個物品的編號是一而非零）。已知 $1 \le n \le 100$、$1 \le B \le 10000$、$1 \le m \le 10$、$1 \le w_i \le 100$、$1 \le v_i \le 100$、$1 \le q_j \le n$。每一列的任兩個相鄰的整數間以一個空白字元隔開。

請依題目指示，先輸出 m 個解中可行解的數量，以及可行解中最大的總價值（如果所有的解都不可行則輸出 0），兩個數字之間用一個逗點隔開。舉例來說，如果輸入是

```
4 9 3
3 4 2 3
5 4 1 2
```

```
3 1 2 3
3 2 1 4
2 3 4
```

則輸出應該是

```
2,10
```

如果輸入是

```
4 9 2
3 4 2 3
5 4 1 2
4 1 2 3 4
3 2 1 4
```

則輸出應該是

```
0,0
```

你上傳的原始碼裡應該包含什麼

你的 .cpp 原始碼檔案裡面應該包含讀取測試資料、做運算，以及輸出答案的 C++ 程式碼。當然，你應該寫適當的註解。針對這個題目，你可以使用任何方法。

評分原則

這一題的所有分數都根據程式運算的正確性給分。PDOGS 會編譯並執行你的程式、輸入測試資料，並檢查輸出的答案的正確性。一筆測試資料佔 2 分。

第三題

（50 分）承前兩題，現在我們將指定一個貪婪演算法（greedy algorithm），請你實作這個演算法以求得一個可能還不錯的可行解。請依照以下演算法來挑選適當的物品，以達到最大化總價值的目標：

> 一開始背包裡是空的，也就是什麼物品都沒有被選取。接著我們進行數輪挑選，在每一輪中找出還沒被選且選取之後不會超過耐重度上限的所有物品，從中挑選價值最高的把它放入背包中。如果同時有兩個以上的物品價值相同時，選擇編號較小的那一個。如果任何未選物品放入背包後都會超過耐重度上限，就結束運算，否則就進行下一輪。

上面這段文字雖然不失為一個正確的演算法敘述，但因為不是以程式結構描述，讀起來就是有點模糊。此外，要從它直接變成程式，對初學者來說也不免有些挑戰。此時如果把它改寫成如下的 pseudocode，應該會有幫助：

```
初始化 x[1] = x[2] = ... = x[n] = 0
while(true)
{
  令 K 為所有還沒被選且選取之後不會超過耐重度上限的所有物品之集合
  if(K 是空集合)
    break
  else
  {
    從 K 中挑出價值最高的物品 i，若平手則挑編號最小的
    令 x[i] = 1
  }
}
return x
```

你可以看到 pseudocode 的形式是很自由的，只要對你寫程式有幫助就好。你可以繼續修改這個 pseudocode 去往裡面加入更多細節，直到你覺得整個程式都架構好了，就可以開始寫程式了[3]！

讓我們多舉一個例子來說明這個演算法。假設我們有 5 件物品，重量依序分別是 5、2、2、2、6，而價值依序分別是 3、3、1、2、3，背包耐重度上限為 10。以上述演算法執行此例，我們會得到：

1. 一開始背包裡是空的，也就是什麼物品都沒有被選取。
2. 接著找出價值最高，同時選取之後又不會超過耐重度上限的物品，把它放入背包中。符合條件的是價值最高的 1 號、2 號以及 5 號物品，他們的價值都是最高的 3，但基於要選取編號最小的，我們會在第一輪選擇 1 號物品放入背包，總價值變成 3，而總重量變成 5，還沒有超過耐重度上限 10，因此再選取下一個物品[4]。
3. 未選取的四項物品中，價值最高，同時選取之後又不會超過耐重度上限的物品為 2 號以及 5 號物品，同樣因為平手時要選取編號小的，因此我們會在第二輪選擇 2 號物品放入背包，總價值變成 $3+3=6$，而總重量變成 $5+2=7$，還沒有超過耐重度上限 10，因此再選取下一個物品。
4. 未選取的三項物品中，價值最高，同時選取之後又不會超過耐重度上限的物品為 4 號物品（價值較高的 5 號物品因為超過耐重度上限，所以不列入考慮）。因此我們會在第三輪選擇 4 號物品放入背包，總價值變成 $6+2=8$，而總重量變成 $7+2=9$。
5. 未選取的兩項物品中，重量分別是 2 以及 6，放入任何一項都會超過重量上限。因此我們結束演算。透過這套演算法，我們會選擇將 1、2、4 號物品放入背包中，並且獲得總價值為 3+3+2 = 8。

請完成以上演算法實作，並依照挑選的順序輸出被挑選的物品編號，以及所有被選取物品的總價值為何。

[3]搞不好你可以把 pseudocode 貼給 ChatGPT，請他幫你寫 C++！如果這樣能完成作業，我認為你的表現也是合格的。

[4]你可能已經發現，此時好像沒有理由挑編號小的，應該要挑重量小的才對。不管怎樣，現在我們沒有要讓你學習設計好的演算法，你只要能實作指定的演算法就好了。如果你有興趣，可以自己把演算法改成選重量小的，然後自己生成許多測試資料去測試，看看選重量小的一般來說可以提升多少成效。這應該也是很有趣的！

輸入輸出格式

系統會提供一共 15 組測試資料，每組測試資料裝在一個檔案裡。在每個檔案中會有三列，第一列裝著兩個正整數，依序是 n、B，代表物品數量以及耐重度上限；第二列存了 n 個非負整數，依序是 w_1、w_2 直到 w_n，分別代表每一個物品的重量；第三列存了 n 個非負整數，依序是 v_1、v_2 直到 v_n，分別代表每一個物品的價值。已知 $1 \leq n \leq 100$、$1 \leq B \leq 10000$、$1 \leq w_i \leq 100$、$1 \leq v_i \leq 100$。每一列的任兩個相鄰的整數間以一個空白字元隔開。

請依題目要求與指示，實作演算法，並依照挑選之順序依序輸出被挑選之物品的編號，編號之間以一個逗點隔開，最後加一個分號，再輸出所有選取物品的總價值；如果所有物品都無法被放進背包（因為重量都超過耐重度上限）則印出分號後面接一個 0。舉例來說，如果輸入是

```
4 9
4 3 2 3
4 5 3 2
```

則輸出應該是

```
2,1,3;12
```

如果輸入是

```
5 10
5 2 2 2 6
3 3 1 2 3
```

則輸出應該是

```
1,2,4;8
```

如果輸入是

```
5 1
5 2 2 2 6
3 3 2 2 3
```

則輸出應該是

```
;0
```

你上傳的原始碼裡應該包含什麼

你的 .cpp 原始碼檔案裡面應該包含讀取測試資料、做運算，以及輸出答案的 C++ 程式碼。當然，你應該寫適當的註解。針對這個題目，你***不可以***使用上課沒有教過的方法：

- 確定可以使用的語法包含 `if-else`、`for`、`while`、陣列、函數、`<climits>` 裡面所有的東西、`<iomanip>` 裡面所有的東西、`<cmath>` 裡面的 `abs()` 和 `sqrt()`、`sizeof()`、`static_cast()`、constants 等。

- 確定不可以使用的語法包含 `printf`、`scanf`、`max`、`min`、`<cmath>` 裡面除了 `abs()` 和 `sqrt()` 以外的函數、動態配置記憶體等等。

請注意正面表列的固然是都確定可以用，但沒有被負面表列的不表示可以用喔！

此外，請幫你的程式寫適當的函數，例如你第一題已經寫好的函數，或者你自己設計的函數，但總之你的程式應該要藉由使用函數來實現理想的程式結構。

評分原則

- 這一題的其中 30 分會根據程式運算的正確性給分。PDOGS 會編譯並執行你的程式、輸入測試資料，並檢查輸出的答案的正確性。一筆測試資料佔 2 分。
- 這一題的其中 20 分會根據你所寫的程式的品質來給分。助教會打開你的程式碼並檢閱你的程式的結構、運算邏輯、可讀性（包含排版、變數命名、註解等等）、可擴充性、模組化程度，以及是否使用了還沒教過的語法。請寫一個「好」的程式吧！

第四題

（20 分）承上題，但這次我們將指定另一個演算法：首先，選取所有的物品，如果沒有超重那就結束演算，如果超過耐重度則開始逐步移除物品。在每一輪中我們找出該移除哪一個物品，可以讓背包內的物品不超過耐重度上限，同時又可以保留最大的價值，也就是拿出價值最小的物品。如果同時有兩個以上的物品價值相同，請選擇編號較小的那個。移除該物品之後，因為已經獲得一項可行解，即可結束迴圈。如果此輪不論拿出哪一個物品，都仍然會超過耐重度上限，請移除價值最低的物品，如果有數個物品的價值都最低則從中選擇編號最小者，接著進行下一輪篩選。

跟之前一樣，讓我們試著寫個對演算法敘述更精確的 pseudocode：

```
初始化 x[1] = x[2] = ... = x[n] = 1
if(沒超重)
  return x
while(true)
{
  令 K 為所有還沒被移除且移除後能使總重量不超過耐重度上限的所有物品之集合
  if(K 是空集合)
  {
    從所有還沒被移除的物品中挑出價值最低的物品 i，若平手則為編號最小的
    令 x[i] = 0
  }
  else
  {
    從 K 中挑出價值最低的物品 i，若平手則為編號最小的
    令 x[i] = 0
    break
  }
```

```
}
return x
```

舉例來說，假設我們有 5 件物品，重量依序分別是 5、2、2、2、6，而價值依序分別是 3、3、1、2、3，背包耐重度上限為 10。對此例執行演算法的步驟如下：

1. 一開始將五項物品全部塞進背包裡，此時總重量為 5+2+2+2+6 = 17，總價值為 3+3+1+2+3 = 12。
2. 接著找出我們該移除哪一個物品，可以讓背包內的物品不超過耐重度上限，同時又可以保留最大的價值，也就是拿出價值最小的物品。在此例中，拿出任何一個物品，都無法讓背包內總重量符合耐重度上限，因此我們直接尋找價值最小的物品，也就是說第一個被移除的物品是價值最低的 3 號物品。我們會在第一輪選擇 3 號物品拿出背包，總價值變成 $12 - 1 = 11$，而總重量變成 $17 - 2 = 15$，仍然超過耐重度上限 10，因此再選取下一個物品。
3. 未選取的四項物品中，價值最小，同時移除之後就可以滿足耐重度上限的物品為 1 號以及 5 號物品，此時我們應該選擇其中價值較低者，因為平手所以選取編號小的 1 號物品拿出背包，總價值變成 $11 - 3 = 8$，而總重量變成 $15 - 5 = 10$，符合耐重度限制，演算結束。透過這套演算法，我們會選擇將 2、4、5 號物品留在背包中，並且獲得總價值為 $3 + 2 + 3 = 8$。

請完成以上演算法實作，並依照編號由小到大輸出最終被留在背包中的物品編號，以及所有被選取物品的總價值為何。

輸入輸出格式

系統會提供一共 10 組測試資料，每組測試資料裝在一個檔案裡。輸入格式與第三題都完全相同。請依題目要求與指示實作演算法，並依照編號由小到大輸出最終被留在背包中的物品編號，編號之間以一個逗點隔開，最後加一個分號，再輸出所有選取物品的總價值；如果所有物品都無法被放進背包（因為重量都超過耐重度上限）則印出分號後面接一個 0。舉例來說，如果輸入是

```
4 9
3 4 2 3
5 4 3 2
```

則輸出應該是

```
1,2,3;12
```

如果輸入是

```
5 10
5 2 2 2 6
3 3 1 2 3
```

則輸出應該是

```
2,4,5;8
```

如果輸入是

```
5 1
5 2 2 2 6
3 3 2 2 3
```

則輸出應該是

```
;0
```

你上傳的原始碼裡應該包含什麼

你的 .cpp 原始碼檔案裡面應該包含讀取測試資料、做運算，以及輸出答案的 C++ 程式碼。當然，你應該寫適當的註解。針對這個題目，你可以使用任何方法。

評分原則

這一題的所有分數都根據程式運算的正確性給分。PDOGS 會編譯並執行你的程式、輸入測試資料，並檢查輸出的答案的正確性。一筆測試資料佔 2 分。