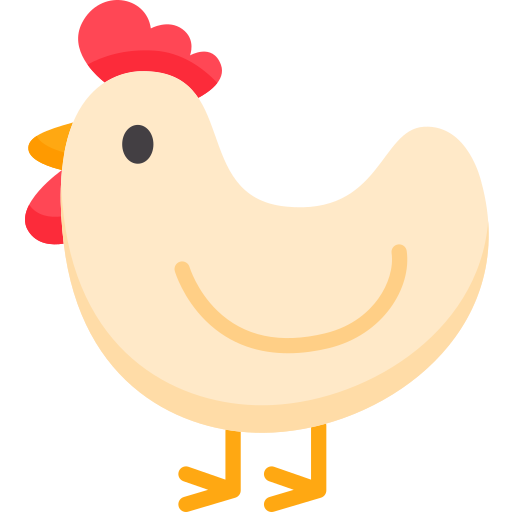
**國立鳳新高中 112 年資訊學科競賽校內賽**





****

裁判雞

**注意事項**

1. 本次競賽總共有6題，使用cms自動評分，請上傳原始程式碼(\*.cpp/\*.c)。
2. 考試時間共150分鐘，競賽結束後將無法再上傳程式碼，比賽期間請勿交談討論。
3. 請勿在程式中加入 system("pause")等指令。
4. 每次上傳之間必須間隔60秒，每題以最後一次評分結果計分。

**各題概況**

|  |  |  |  |
| --- | --- | --- | --- |
| **題目名稱** | **時間限制** | **記憶體限制** | **評分方式** |
| 美味布丁 | 0.1s | 64MBytes | Tolerant (寬鬆比對) |
| 電梯巨星 | 1s | 64MBytes | Tolerant (寬鬆比對) |
| 資料驗證 | 1s | 64MBytes | Strictly (嚴格比對) |
| 市集採買 | 0.5s | 64MBytes | Tolerant (寬鬆比對) |
| 滿頭問號 | 1s | 64MBytes | Special (自訂比對) |
| 草莓大盜 | 1s~2s | 64MBytes | Tolerant (寬鬆比對) |

**A. 美味布丁**

　　日頭赤炎炎，在山頭奔波的烏骨雞倍感炎熱，感覺都快中暑了！「好想吃一口沁涼的布丁！」烏骨雞說。因此，在山腳的巨星廚師雞決定做一道簡單的布丁來滿足烏骨雞。廚師雞找到的布丁食譜如下：

　　　　「

　　　　　　鮮奶280克

　　　　　　雞蛋液120克

　　　　　　鮮奶油30克

　　　　　　砂糖30克

　　　　　　焦糖5克

　　　　」

　　雖然廚師雞做的布丁超級完美，但牠還是持續更新食譜比例，追求極致美味的布丁。礙於食材有限，廚師雞想盡可能用現有的食材，做多一點布丁，但牠做的新食譜太多，沒辦法一個一個算，因此想用程式來算出牠最多可以做出多少克的布丁。

　　以牠找到的食譜為例，化為最簡整數比的食譜比例為鮮奶：雞蛋液：鮮奶油：砂糖：焦糖=56：24：6：6：1，假設牠今天有鮮奶336克、雞蛋液144克、鮮奶油36克、砂糖36克、焦糖5克，那牠最多能做出465克的布丁。

　　今天廚師雞依序給你5個食材的比例，還有5個食材現在有的克數，其中因為食材一定是最新鮮的，所以食材現有的克數一定是食材比例的正整數倍，請你幫牠寫一個程式，算出牠最多可以做出多少克的布丁。

**輸入格式**

　　第一行有五個正整數，分別代表鮮奶、雞蛋液、鮮奶油、砂糖、焦糖的比例，第二行有五個正整數，分別代表鮮奶、雞蛋液、鮮奶油、砂糖、焦糖的現有克數。其中每個數值都不超過105，且食材現有的克數一定是食材比例的正整數倍。

**輸出格式**

　　輸出只有一個整數，代表最多能做出的布丁克數。

**測試資料**

|  |  |
| --- | --- |
| 輸入範例1  56 24 6 6 1  336 144 36 36 5 | 輸出範例1  465 |
| 輸入範例2  23 10 1 5 6  46 50 3 80 30 | 輸出範例2  90 |

**測試資料說明**

　　範例測資1：在這個範例中，只有焦糖為食材比例的5倍，其餘食材皆為食材比例的6倍，因此焦糖是最快耗盡的，故整體比例為鮮奶：雞蛋液：鮮奶油：砂糖：焦糖=280：120：30：30：5。最多做出465克的布丁。

　　範例測資2：在這個範例中，鮮奶為最快耗盡的，整體比例為鮮奶：雞蛋液：鮮奶油：砂糖：焦糖=46：20：2：10：12。最多做出90克的布丁。

**配分**

|  |  |  |  |
| --- | --- | --- | --- |
| **記憶體限制** | 64MBytes | **評分方式** | Tolerant (寬鬆比對) |
| **編號** | **配分** | **時間限制** | **敘述** |
| #0~#3 | 4% | 0.1s | 所有食材都會剛好用完 |
| #4~#99 | 96% | 0.1s | 無特別限制 |

**B. 電梯巨星**

　　終端雞大樓是個非常高聳的大樓，因為實在是太高了，因此大樓裡有一座電梯，能夠從1樓直達100000樓。有一天，巨星廚師雞在終端機電梯裡亂按按鈕，造成電梯不停上上下下，這時終端雞大樓的雞住戶都知道是廚師雞來了，許多雞都迫不及待地等在電梯前，祈禱電梯會停靠在自己所在的樓層。慶幸的是，電梯在上升或下降的過程中，經過的每個樓層都會停靠，因此大大增加了粉絲進入電梯的機會。

　　廚師雞為了提前準備足夠的布丁給粉絲們，於是透過偵查雞調查哪些樓層有雞在電梯門前等候，以此確定有多少雞能夠進入電梯拜訪牠。請你幫忙寫一個程式，計算有多少雞能夠進入電梯拜訪牠。

**輸入格式**

　　輸入共有三行，第一行有兩個整數n(1≤n≤100), m(1≤m≤105)，n代表電梯停靠的樓層數量，m代表雞的數量。第二行有n個整數ni(1≤ni≤105)，代表電梯依序抵達的樓層。第三行有m個整數mi(1≤mi≤105)，代表第i隻雞所在的樓層。

**輸出格式**

　　輸出只有一行，包含一個整數n，代表總共有n隻雞可以進入電梯。

**測試資料**

|  |  |
| --- | --- |
| 輸入範例1  2 5  1 10  2 5 9 13 17 | 輸出範例1  3 |
| 輸入範例2  2 5  10 1  2 5 9 13 17 | 輸出範例2  3 |
| 輸入範例3  5 5  10 1 15 12 16  2 5 9 13 17 | 輸出範例3  4 |
| 輸入範例4  5 5  6 7 5 6 4  1 8 3 10 15 | 輸出範例4  0 |

**測試資料說明**

　　範例測資1中，電梯會從1樓上升到10樓，所以在1~10樓的雞都能夠進入電梯，因此總共有3隻雞，分別是在2、5、9樓的雞。

　　範例測資2中，電梯從10樓下降到1樓，同樣是在1~10樓的雞能夠進入電梯，因此答案與範例測資1相同。

　　範例測資3中，電梯從10樓下降到1樓時，2、5、9樓的雞能夠進入電梯；電梯從1樓上升到15樓時，13樓的雞能夠進入電梯，因此總共有4隻雞能夠進入電梯，分別是2、5、9、13樓的雞。

　　範例測資4中，沒有雞可以進入電梯，因此輸出0。

**配分**

|  |  |  |  |
| --- | --- | --- | --- |
| **記憶體限制** | 64MBytes | **評分方式** | Tolerant (寬鬆比對) |
| **編號** | **配分** | **時間限制** | **敘述** |
| #0~#4 | 10% | 1s | n=2且m≤100，且電梯只會上樓 |
| #5~#9 | 10% | 1s | n=2 |
| #10~#29 | 80% | 1s | 無特別限制 |

**C. 資料驗證**

　　放山雞在進行資料傳輸或是提取資料時，為了確保資料的正確性，因此會進行資料驗證。其中一種資料驗證方式，是同位位元。同位位元驗證資料的方式分為兩種，奇核對位元以及偶核對位元。

　　以奇核對位元為例，給定一組資料中，若資料中1的個數為奇數，則補一個0，反之若資料中1的個數為偶數，則補一個1。以偶核對位元為例，給定一組資料中，若資料中1的個數為奇數，則補一個1，若資料中1的個數為偶數，則補一個0。

　　舉例來說，以1組7位元的資料組為例，對於「1000101」，補上奇核對位元為「10001010」，補上偶核對位元則為「10001011」，若驗證時的資料組，無法符合以上規則，我們就稱資料毀損。

　　今天給定n組m位元加1位同位位元資料，在k=1時採用奇核對位元，k=0時採用偶核對位元，請對驗證正確的資料組輸出0，資料毀損的資料組輸出1，其中不含任何空白以及換行。

　　請依據以上規則，幫放山雞作出一套能驗證資料的程式。

**輸入格式**

　　第一行有三個正整數n,m,k，其中n為資料組組數且1≤n≤5000，m為一組資料組的位元數(不包含同位位元)且1≤m≤10000，k必為1或0，k=1時採用奇核對位元，k=0時採用偶核對位元，第二行有n×(m+1)個數字，而其中的數字必為0或1。

**輸出格式**

　　輸出只有一行，共有n個數字，且每一個數字必為0或1，其中不得包含任何空白以及換行。

**測試資料**

|  |  |
| --- | --- |
| 輸入範例1  1 7 1  10001010 | 輸出範例1  0 |
| 輸入範例2  1 7 0  10001010 | 輸出範例2  1 |
| 輸入範例3  5 3 1  10101011101010111010 | 輸出範例3  10101 |
| 輸入範例4  3 6 0  100101011011011111110 | 輸出範例4  110 |

**測試資料說明**

　　範例測資1、2請見題目敘述。

　　範例測資3中，「1010」的資料組為資料毀損的資料，輸出1，「1011」的資料組為驗證正確的資料，輸出0。

　　範例測資4中，「1001010」及「1101101」的資料組為資料毀損的資料，輸出1，「1111110」的資料組為驗證正確的資料，輸出0。

**配分**

|  |  |  |  |
| --- | --- | --- | --- |
| **記憶體限制** | 64MBytes | **評分方式** | Strictly (嚴格比對) |
| **編號** | **配分** | **時間限制** | **敘述** |
| #0~#1 | 6% | 1s | n=1,1≤m≤100 |
| #2~#4 | 9% | 1s | 1≤n≤100,1≤m≤100 |
| #5~#9 | 33% | 1s | 1≤n≤1000 |
| #10~#19 | 52% | 1s | 無特別限制 |

**D. 市集採買**

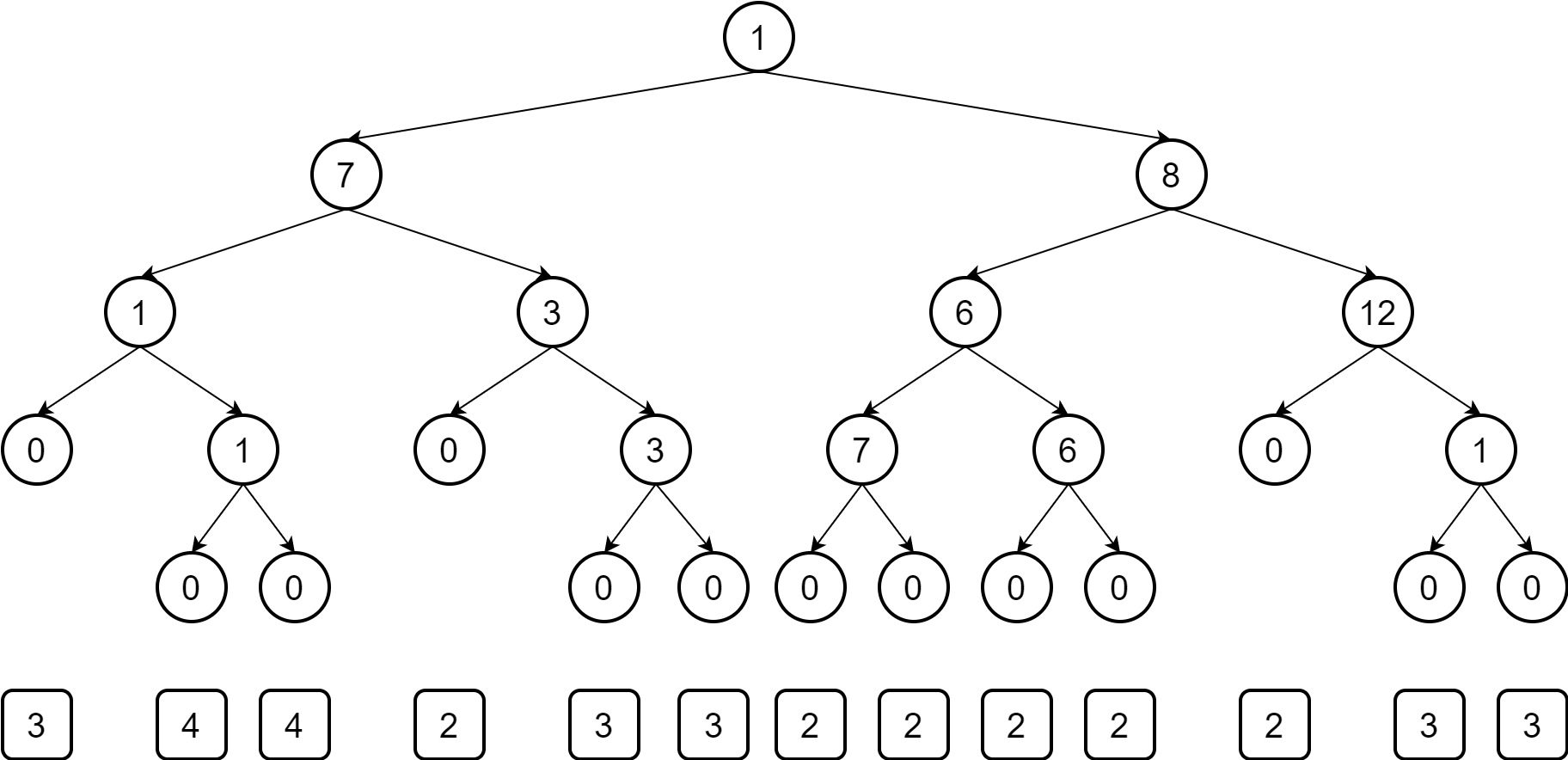
　　終端雞市集，是一個以二元樹為原型，打造出的市集，強大的設計理念，吸引許多攤販進駐。所有攤販都被安排在除了葉節點以外的節點，而葉節點只會有「已經到盡頭」的告示牌，入口則安排在根結點。攤販們被規定只能在攤位上賣一件物品，並且需要標明這件物品的大小，以便大家放心採買。

烏骨雞和放山雞放學後，決定去終端雞市集採買，烏骨雞想要盡可能在不超過背包容量下，買到最多物品，但放山雞太懶惰，不想逛完整個市集，最後，兩雞達成協議，只會逛市集的其中一條路線，然後原路折返。

　　即便如此，烏骨雞還是想盡可能買到最多物品，因此牠想設計一個程式，輸入背包容量後，再以前序輸入市集攤販所賣物品之大小，就能獲得一條能買最多物品的路線。但因為可能同時存在多條路線，烏骨雞又有選擇困難，因此輸出永遠只會輸出最右邊那條。

　　以終端雞市集為例，假設烏骨雞背包容量為10，以前序輸入市集攤販所賣物品之大小為「1 7 1 0 1 0 0 3 0 3 0 0 8 6 7 0 0 6 0 0 12 0 1 0 0」，其中0為「已經到盡頭」的告示牌。而樹的圖如下圖所示，圓形內數字代表攤販所賣物品之大小，最底下的矩形內數字代表該條路在不超過背包容量下，所能買到最多物品數量，而輸出要選擇最多物品又最右邊的路線，所以輸出為「1 7 1 1 0」。

　　由於烏骨雞能力不足，因此請你依所給條件，幫烏骨雞寫一個程式來達成目的。

****

**輸入格式**

　　第一行有一個不超過106的正整數，代表烏骨雞的背包容量，第二行有 2n個攤販所賣物品之大小，n不超過20，每個攤販所賣物品之大小為一個不超過105的正整數。

**輸出格式**

　　輸出只有一行，為一條能買最多物品且最右邊的路線。

**特別測資限制**

1. 二元樹深度不超過20層。

**測試資料**

|  |  |
| --- | --- |
| 輸入範例1  7  3 4 0 0 5 0 0 | 輸出範例1  3 4 0 |
| 輸入範例2  10  1 7 1 0 1 0 0 3 0 3 0 0 8 6 7 0 0 6 0 0 12 0 1 0 0 | 輸出範例2  1 7 1 1 0 |

**測試資料說明**

　　範例測資1中，總共有四條路徑，由左到右分別為「3 4 0」「3 4 0」「3 5 0」「3 5 0」，前兩條路徑能買到最多物品數量為2，後兩條為1，又選擇最右邊的路徑，故輸出「3 4 0」。

　　範例測資2請見題目敘述以及圖片。

**配分**

|  |  |  |  |
| --- | --- | --- | --- |
| **記憶體限制** | 64MBytes | **評分方式** | Tolerant (寬鬆比對) |
| **編號** | **配分** | **時間限制** | **敘述** |
| #0~#10 | 11% | 0.5s | 二元樹深度不超過5層 |
| #11~#29 | 19% | 0.5s | 二元樹深度不超過10層 |
| #30~#52 | 23% | 0.5s | 二元樹深度不超過15層 |
| #53~#99 | 47% | 0.5s | 無特別限制 |

**E. 滿頭問號**

　　烏骨雞使用通訊軟體與放山雞聊天時，總是會傳一些意義不明的訊息，讓放山雞滿頭問號。於是，放山雞決定在頭上的問號消失之前，將問號們打在聊天室傳給烏骨雞，藉此讓烏骨雞認知到自己的訊息有多麼令人不解。

　　放山雞的鍵盤上有四個功能鍵，分別是：

**「**

　　　　　　A 輸入一個問號

　　　　　　D 刪除一個問號

　　　　　　C 複製

　　　　　　V 貼上

**」**

　　其中C指的是複製當前所有的問號。

　　請你寫一個程式，幫助放山雞找出最少步驟打出特定問號數量的方法。

**輸入格式**

　　輸入只有一行，含有一個整數n(1≤n≤106)，代表放山雞要在聊天室中打出的問號數量。

**輸出格式**

　　請輸出使用最少步驟打出n個問號的方法，請勿輸出空白或其他與題目無關之文字。注意，答案可能不只有一種，輸出其中一種答案即可。

**測試資料**

|  |  |
| --- | --- |
| 輸入範例1  1 | 輸出範例1  A |
| 輸入範例2  9 | 輸出範例2  AAACVV |
| 輸入範例3  10 | 輸出範例3  AAACVVA |
| 輸入範例4  179 | 輸出範例4  AAACVVVVCVVCVCVD |

**測試資料說明**

　　此說明僅說明範例輸出的結果。

　　範例測資1：題目要求打出1個問號，因此只需按下A鍵即可。

　　範例測資2：先用A鍵打出3個問號，再用C鍵複製，然後用V鍵貼上2次。

　　範例測資3：先以範例測資2的方法打出9個問號，再用A鍵補1個問號。

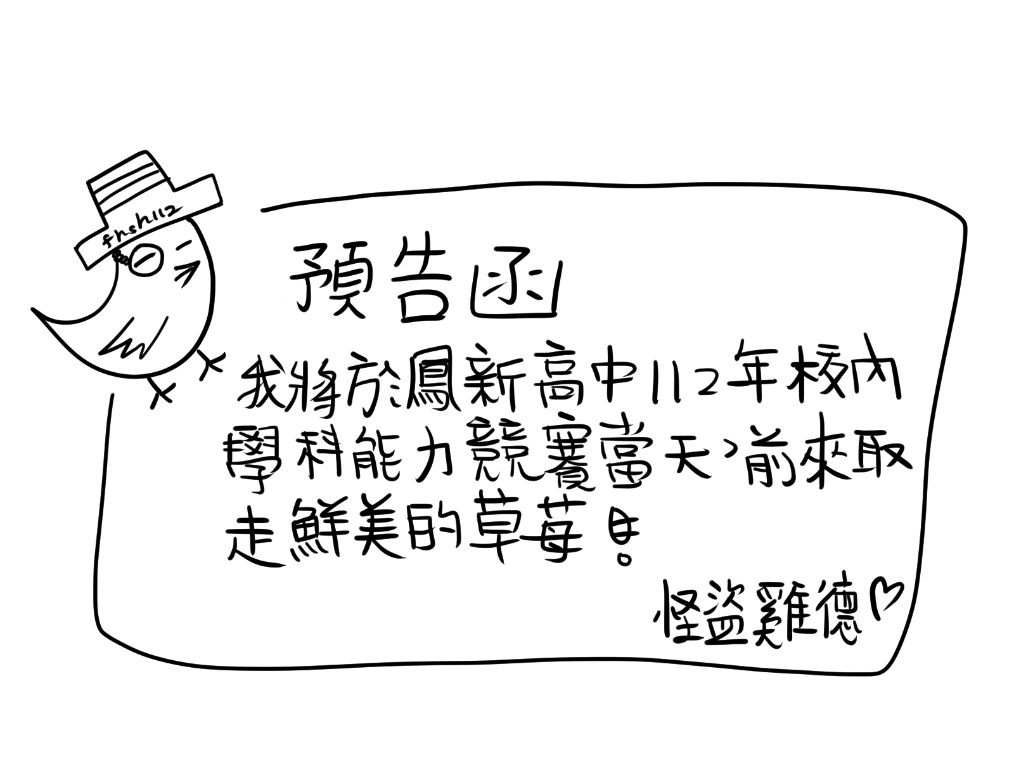
　　範例測資4：用A鍵打出3個問號，用C鍵複製，V鍵貼上4次，得到15個問號，C鍵複製，V鍵貼上2次，得到45個問號，C鍵複製，V鍵貼上1次，得到90個問號，C鍵複製，V鍵貼上1次，得到180個問號，最後用D鍵刪掉一個問號。

**配分**

|  |  |  |  |
| --- | --- | --- | --- |
| **記憶體限制** | 64MBytes | **評分方式** | Special (自訂比對) |
| **編號** | **配分** | **時間限制** | **敘述** |
| #0~#4 | 5% | 1s | n≤10 |
| #5~#19 | 15% | 1s | n≤100 |
| #20~#49 | 30% | 1s | n≤10000 |
| #50~#99 | 50% | 1s | 無特別限制 |

**F. 草莓大盜**

　　怪盜雞德是終端雞大樓附近著名的怪盜，牠的行為謹慎，導致每次東西已經被他偷走了，都還不一定會發現。至於會什麼會發現嘛……，因為牠每次行動前都會發出預告函。

　　今天巨星廚師雞無預警收到一封預告函，內容如下：

　　在廚師雞驚嚇的同時，怪盜雞德也在規劃怎麼偷到最多的草莓。廚師雞的草莓園是一座線性草莓園，從入口直接進去，可就太顯眼了。因此，怪盜雞德選定了不同的垂直降落定點，打算透過這些垂直降落定點偷到最多草莓，但是每顆草莓的豐碩程度不同，要是偷到太多乾扁的草莓可就吃虧了。此外，為了避免被發現，怪盜雞德為每個垂直降落定點規定最多行走步數，只要步數一到，就直接飛走，前往下個垂直降落定點(不一定是順序上的下一個)。最厲害的是，怪盜雞德不管是垂直降落那一步(不計入行走步數)、要飛走的那一步或是行走中的任一一步，都可以瞬間偷取草莓。

　　怪盜雞德相當聰明，懂得利用程式來達到最高收穫，牠的程式只要輸入垂直降落定點數量和草莓數量，再依序輸入垂直降落定點位置、最多行走步數和草莓位置、草莓豐碩程度，就可以得出牠這次行動最多可以偷到的草莓總豐碩程度。(怪盜雞德的程式中，預設一單位是牠走的一步，所有位置都是以此單位標記。)

　　請你寫一個功能和怪盜雞德程式一樣的程式，來證明你和牠一樣聰明。

**輸入格式**

　　第一行有兩個正整數n,m，其中n為垂直降落定點數量且1≤n≤106，m為草莓數量且1≤m≤106，接下來有n行，每行分別有垂直降落定點位置ni、最多行走步數nj，

(0≤ni≤107 , 1≤nj≤104)，接下來有m行，每行分別有草莓位置mi、草莓豐碩程度mj，

(0≤mi≤107 , 1≤mj≤109)。

**輸出格式**

　　輸出只有一行，為最多可以偷到的草莓總豐碩程度。

**特別測資限制**

1. 垂直降落定點位置ni不重複。
2. 草莓位置mi不重複。

**測試資料**

|  |  |
| --- | --- |
| 輸入範例1  1 3  4 5  2 5  5 2  6 3 | 輸出範例1  7 |
| 輸入範例2  3 6  12 3  3 3  6 4  0 3  2 2  10 3  15 7  5 10  9 5 | 輸出範例2  25 |

**測試資料說明**

　　範例測資1：從4的位置向左走兩步，取得在位置2豐碩程度為5的草莓，再向右走三步，取得在位置5豐碩程度為2的草莓，到此已經5步，最大豐碩程度為7。亦可先向右1步再向左4步，但若取得在位置6豐碩程度為3的草莓，則會降低整體豐碩程度。

　　範例測資2：從3的位置，向右2步，取得1個草莓，豐碩度共計10。從6的位置，向右3步，再向右1步，取得2顆草莓，豐碩度共計8。從12的位置，向右3步，取得1個草莓，豐碩度共計7。總豐碩度為25。雖然有部分步數沒走完，但以達成最高總豐碩程度為目標。

**配分**

|  |  |  |  |
| --- | --- | --- | --- |
| **記憶體限制** | 64MBytes | **評分方式** | Tolerant (寬鬆比對) |
| **編號** | **配分** | **時間限制** | **敘述** |
| #0~#5 | 30% | 1s | n=1 |
| #6~#7 | 8% | 1s | n=2 |
| #8~#10 | 12% | 1s | 1≤n≤100 |
| #11~#19 | 50% | 2s | 無特別限制 |