Bonus

壹、 實現方法

A. createlist()

1. 用一個迴圈去做 link list，先將目前的節點填入 arr[i]的數值，接著假如 i =0，則代表此為第一個節點，則將其設為 first。
2. 如果不是第一個節點，則將前一個節點指向下一個節點的指標設為現在的節點，最後再將現在的節點設為前一個節點，接著去跑下一個節點。
3. 跑完迴圈後，及代表作好 linklist 了，最後回傳第一個節點

B. printlist()

1. 將第一個節點傳入，並設一個節點變數使其等於 first，如果 first 為 NULL 則代表此為空鏈結串列。
2. 如果不是 first != NULL，則跑一個 while 迴圈，如果當前節點不是 NULL，則代表此並非最後一個節點，就繼續輸出串列的數值，接著將 node 設為下一個節點。
3. 跑完迴圈後，輸出換行，讓排版不會太亂

C. combineList()

1. 傳入兩個陣列的第一個節點，首先找到前面鏈結串列的最後一個節

點，並將其設為 current。

2. 接著將 current 指向下一個節點的指標設為第二個鏈結串列的第一個節點，則完成串列的合併

3. 完成後，則再次輸出第一個鏈結串列，及為合併的串列。

D. insertFirstNode()

1. 首先傳入鏈結串列的第一個節點，並且輸入你想插入多少節點數。

2. 接著用一個迴圈去插入節點，輸入節點的數值資料（num）

3. 設一個新的節點，將其 data 設為 num，指向下一個的指標設為 first

4. 最後將新的節點設為 first

5. 跑完迴圈後回傳新的 first 節點

6. 完成後，再次輸出鏈結串列，即為插入數值後的鏈結串列。

E. listLength()

1. 首先傳入第一個節點，接著將第一個節點設為現在的節點

2. 設一個 num 值代表節點數，因為第一個已經被設為現在的節點了，因此初始值為 1

3. 當此時的節點不為 NULL 時，則將現在節點指向下一個節點的指標設為下一個節點，且節點數加一

4. 回傳節點數。

F. reverseList()

1. 首先傳入第一個節點，並且設定三個Node 的指標，將 previos 設為 NULL，因為反轉鏈結串列則代表第一個節點將指向ＮＵＬＬ，接著將 current 設為 first，然後將 proceding 設為第二個節點的指標。
2. 如果下一個指標不是 NULL，則代表其並非最後一個節點，首先將現在節點的下一個設為前一個節點，及代表反轉這個陣列，接著將前一個節點改為現節點。
3. 現節點改為下一個節點，下一個節點設為下下個節點
4. 結束迴圈後，此時的節點為最後一個節點，將其指向前一個節點，並將其設為 first，則完成反轉串列
5. 回傳新的 first，並再次輸出出新的鏈結串列。

G. freeList()

1. 傳入第一個節點，並設一個目前節點以及暫存節點，先將目前節點設為第一個節點。
2. 利用迴圈去跑 linklist，首先將暫存節點設為目前節點，並將現節點改為下一個節點，接著是放暫存節點的空間。
3. 當跑完迴圈則代表將所有節點所用的記憶體歸還。

貳、 心得

A. 困難的點：

1. 如何反轉陣列？

   一開始會很簡單的想只要把第一個節點設為最後一個，最後一個設為第一個就好，但是實際上是要將全部串列的方向都轉過去才行，因此想了一陣子如何將一個個節點指向前一個節點，並將第一個節點設為最後節點，指向 NULL

2. 釋放記憶體空間

   因為我是照順序一個個寫的程式，然後每次寫完都會跑一次，因此在做 combinelist 時我在輸出鏈結串列後發現後面會跑出一大串的亂碼，一開始以為是合併函式的問題，但再仔細檢查後發現是因為開始因為有兩個鏈結串列，因此我要釋放兩次記憶體空間，但是因為合併後，只剩一個串列，自然只需要釋放一次，我寫了兩次 freelist 肯定會出問題的。

B. 其他地方的插入節點

1. 在中間插入節點:

   a. 先走訪 linklist，找到要插入節點的位置，利用迴圈找到後回傳，如果沒找到則回傳 NULL。

   b. 接著將此節點傳入插入函式，輸入想要插入的數值

   c. 設一個全新的節點，將找到節點指向的下一個節點改為新節點指向的節點，接著將數值設為此節點的 data，最後將找到的節點指

向新節點

2. 在後面插入節點

   a. 建立一個全新的節點，並將此節點的 data 設為輸入的數值，並將其指標指向 NULL

   b. 利用迴圈找最後一個節點，將其指標指向新節點

C. Singly link list & Double link list

1. Singly link list：含有該節點的數值，並且有一個指標指向下一個節點的位址，最後一個節點指向 NULL。

2. Double link list：含有該節點的數值，並且有一個指標指向前一個節點的位址，還有一個指標指向下一個節點的位址，第一個節點指向前面的指標為 NULL，最後一個指向後面的指標為ＮＵＬＬ

3. 兩者的優缺點：

   a. Singly link list:

      優點為插入和刪除都很方便，且不需要連續的記憶體空間。缺點為需要多儲存下一個 Node 資訊、取出第 N 筆資料速度比陣列慢、需要從 first/ head 開始逐一往下尋找，需要 O(N)的時間複雜度，且反向遍歷 List 不容易實做

   b. Double link list:

      優點為在刪除第 n 筆 Node 前不用先找到第 n-1 筆 Node，且

一個 next 如果並未正確指向下一個節點，不會無法恢復資料，缺點則是要較多空間去儲存前一個節點的資料。

D. Link list vs. Array:

	Link list	Array
Accessing	Sequential	Random
Insertion	O(1)	O(n) in worst case
Deletion	O(1)	O(n) in worst case
Linear Search	O(n)	O(1)
Binary Search	不適用	O(log(n))
Space	O(n)	O(n)
使用時機	1.無法預期資料數量或頻繁變動資料數量時。 2.需要頻繁地新增/刪除資料時。 3.不需要快速查詢資料。	1.快速存取資料時 2.已知資料的數量，如此便不用經常改變陣列大小 3.要求記憶體空間的使用越少越好。