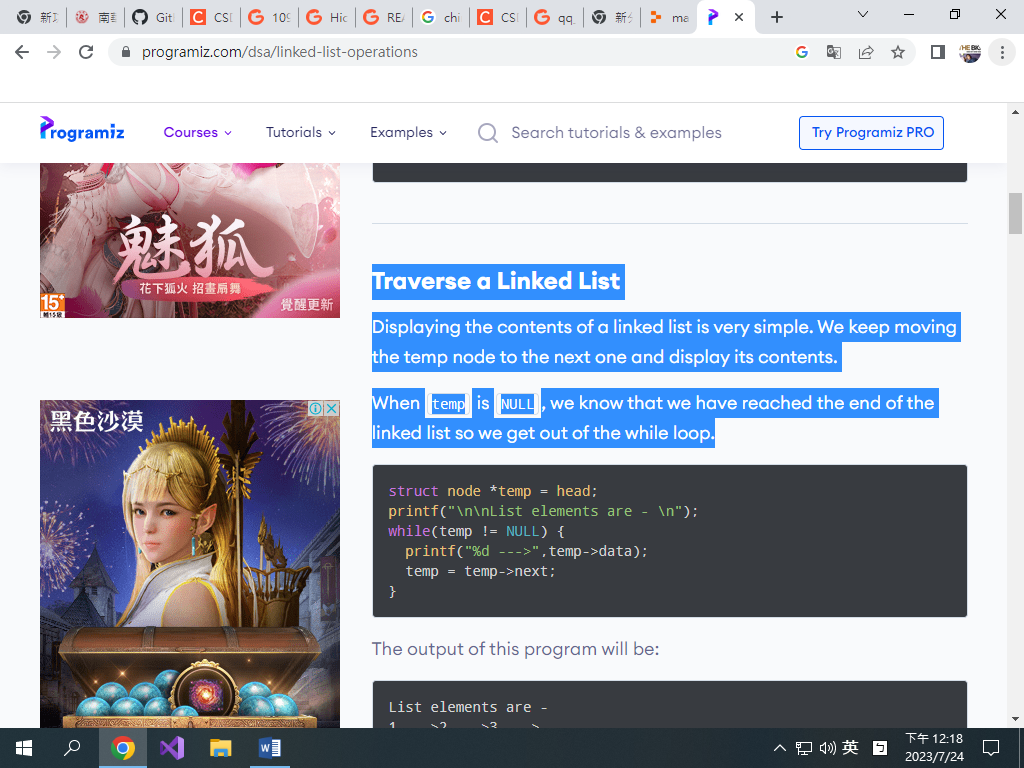
**Summer Class Homework**

Linked list operations: Traverse, insert and delete

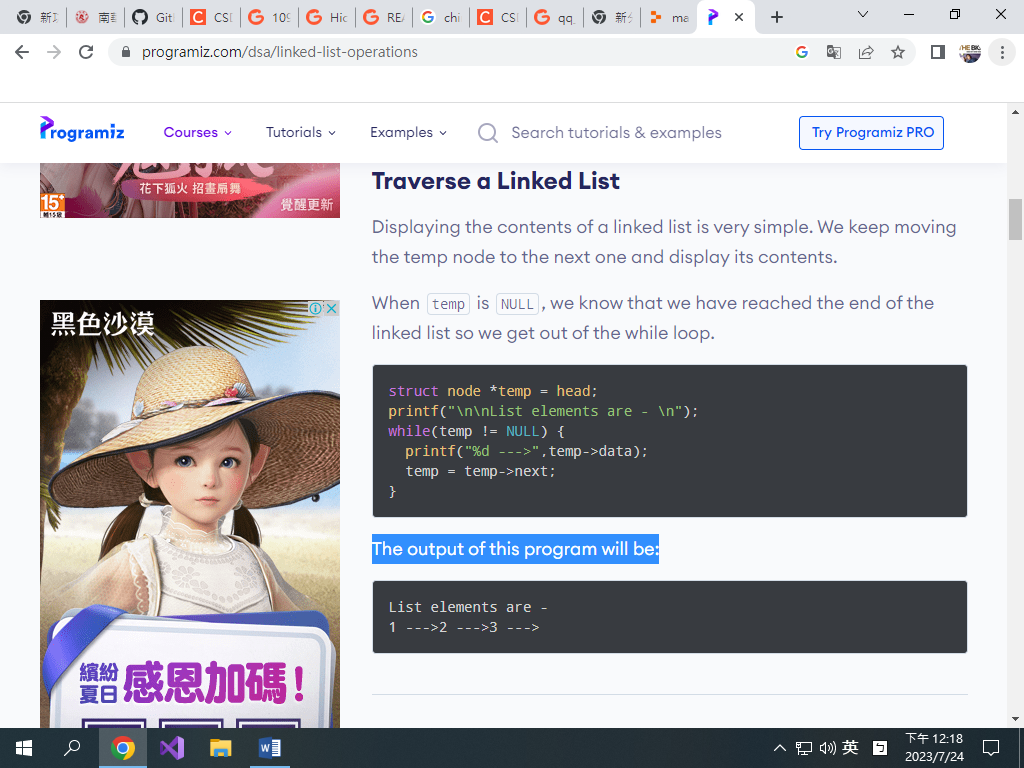
Traverse a Linked List

Displaying the contents of a linked list is very simple. We keep moving the temp node to the next one and display its contents.

When temp is NULL, we know that we have reached the end of the linked list so we get out of the while loop.



The output of this program will be:



### 1.Insert at the beginning

* Allocate memory for new node
* Store data
* Change next of new node to point to head
* Change head to point to recently created node(figure-1)

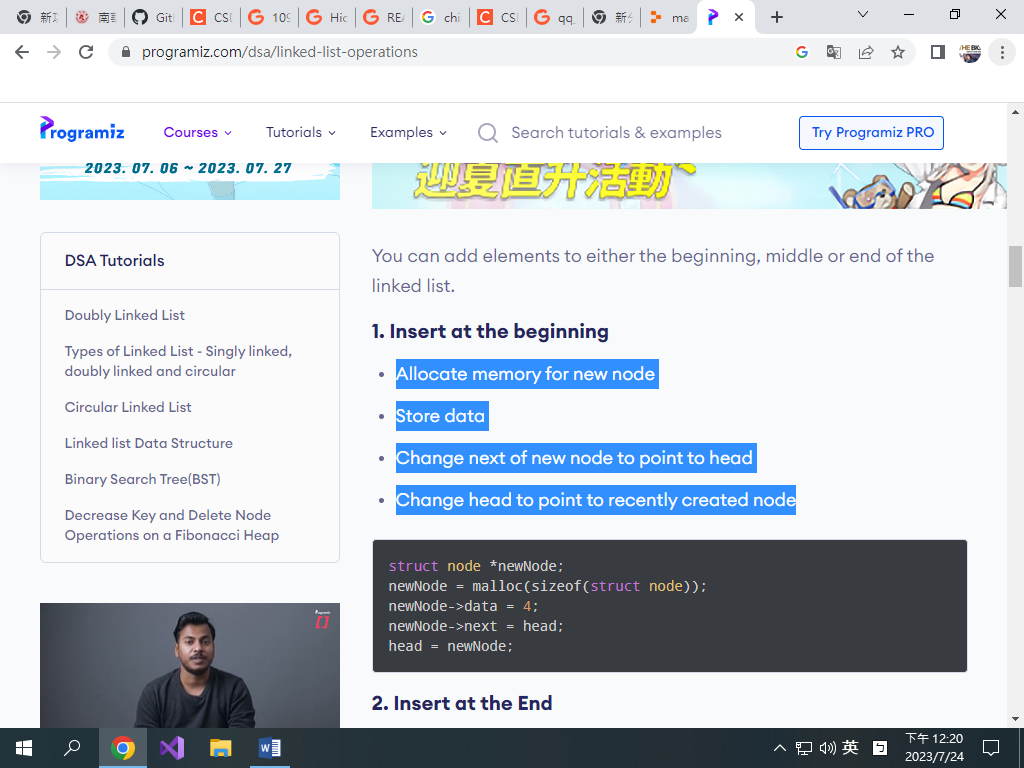


Figure 1: Insert at the beginning

### 2.Insert at the End

* Allocate memory for new node
* Store data
* Traverse to last node
* Change next of last node to recently created node (figure-2)

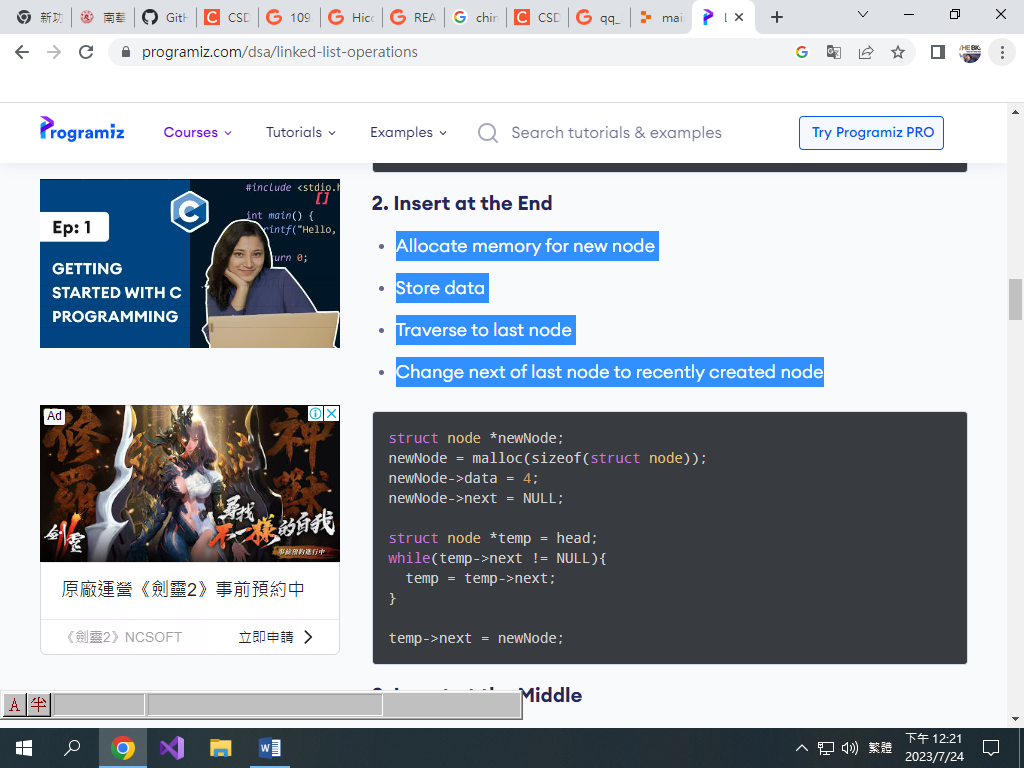


Figure 2: Insert at the End

### 3.Insert at the Middle

* Allocate memory and store data for new node
* Traverse to node just before the required position of new node
* Change next pointers to include new node in between (figure-3)

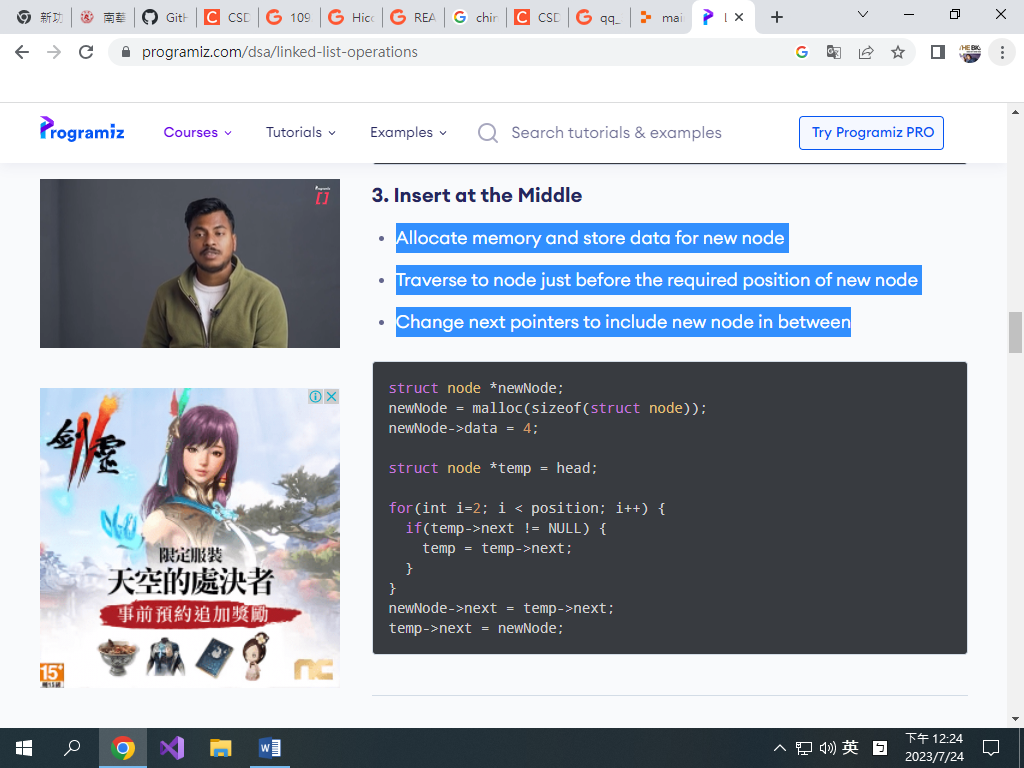


Figure 3: Insert at the Middle

## Delete from a Linked List

You can delete either from the beginning, end or from a particular position.

### 1. Delete from beginning

* Point head to the second node (figure-4)

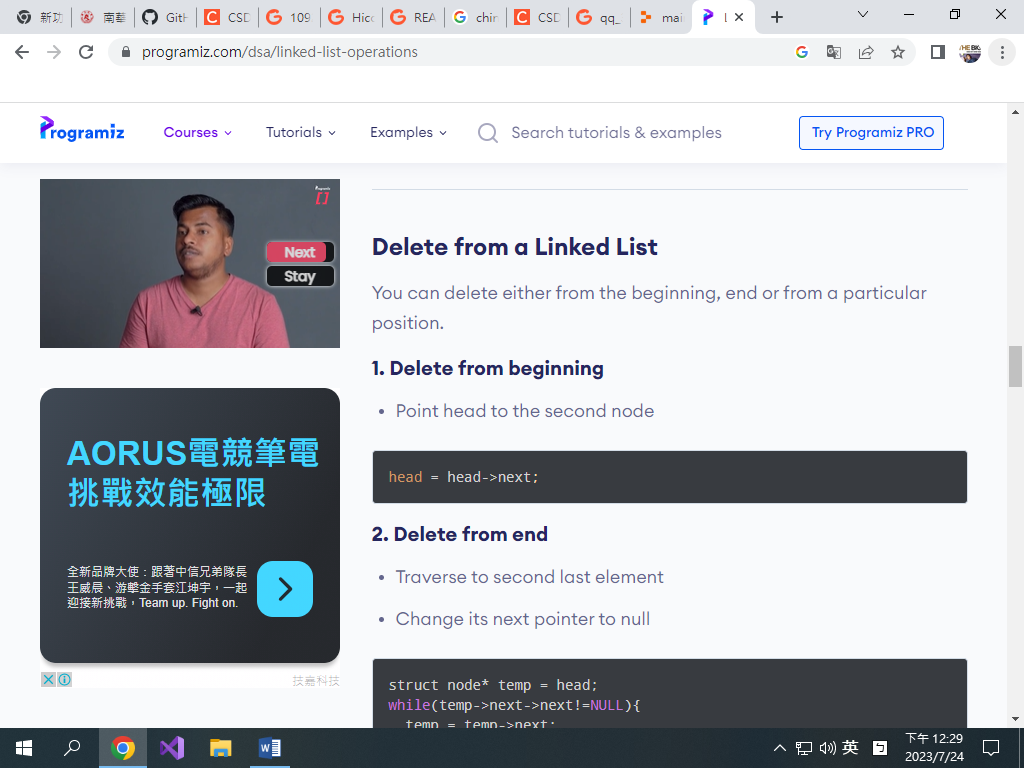


Figure 4: Delete from beginning

### 2. Delete from end

* Traverse to second last element
* Change its next pointer to null (figure-5)

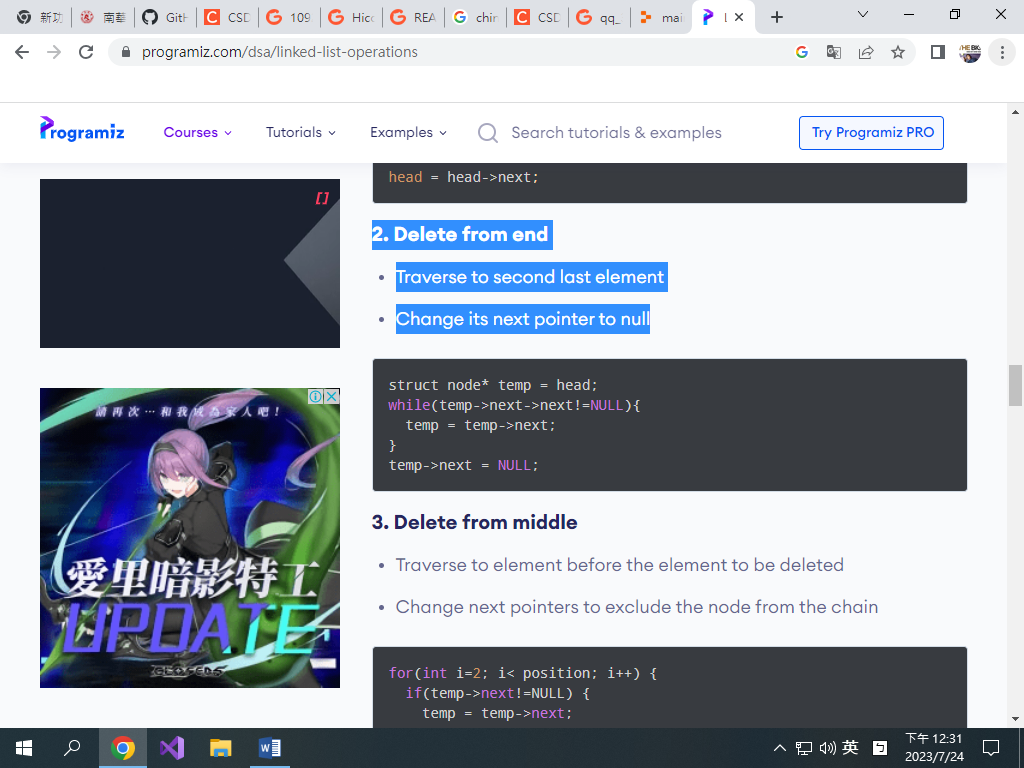


Figure 5: Delete from end

### 3. Delete from middle

* Traverse to element before the element to be deleted
* Change next pointers to exclude the node from the chain(figure-6)

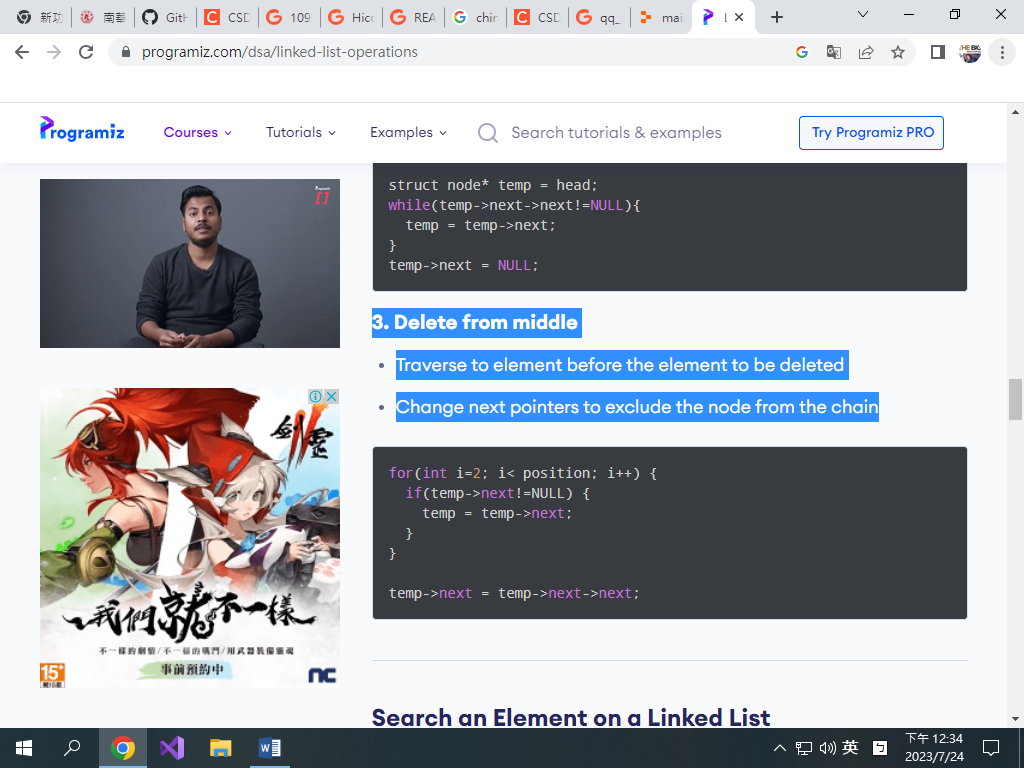


Figure 6: Delete from middle

**Search an Element on a Linked List**

You can search an element on a linked list using a loop using the following steps. We are finding “item” on a linked list.

* Make “head” as the “current” node.
* Run a loop until the “current” node is “NULL” because the last element points to “NULL”.
* In each iteration, check if the key of the node is equal to “item”. If it the key matches the item, return “true” otherwise return “false” (figure-7).

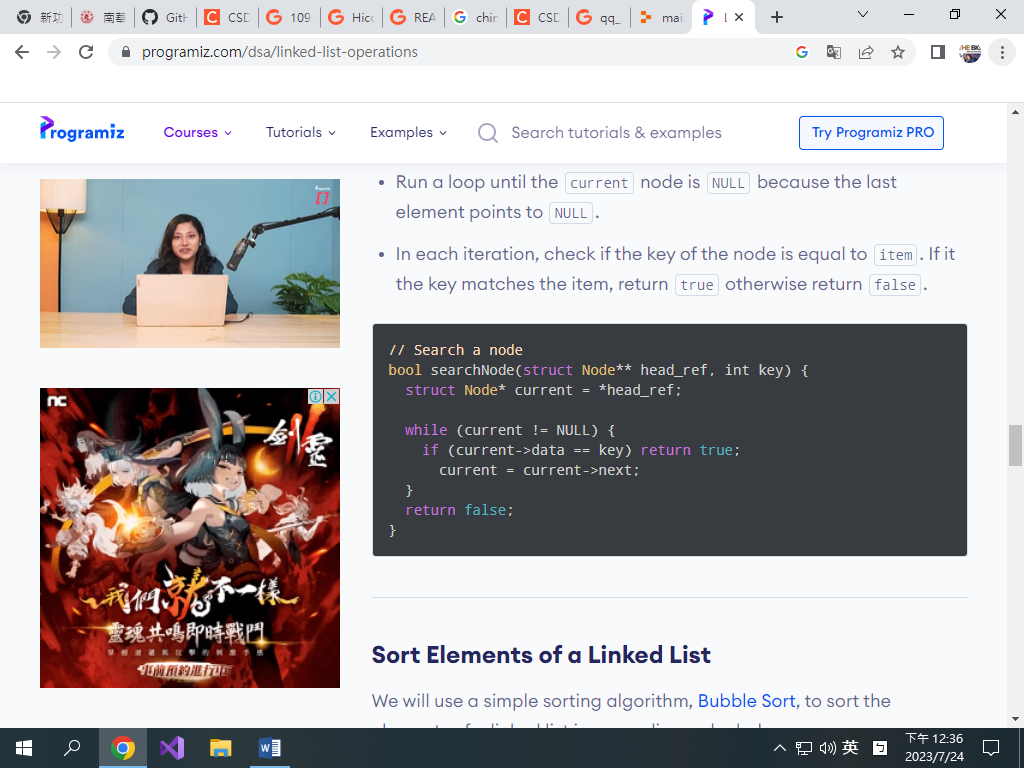


Figure 7: Search an Element on a Linked List

## Sort Elements of a Linked List

We will use a simple sorting algorithm, [Bubble Sort](https://www.programiz.com/dsa/bubble-sort), to sort the elements of a linked list in ascending order below.

1. Make the head as the current node and create another node index for later use.
2. If head is null, return.
3. Else, run a loop till the last node (NULL).
4. In each iteration, follow the following step 5-6.
5. Store the next node of current in index.
6. Check if the data of the current node is greater than the next node. If it is greater, swap current and index.

Check the article on [bubble sort](https://www.programiz.com/dsa/bubble-sort) for better understanding of its working (figure-8).

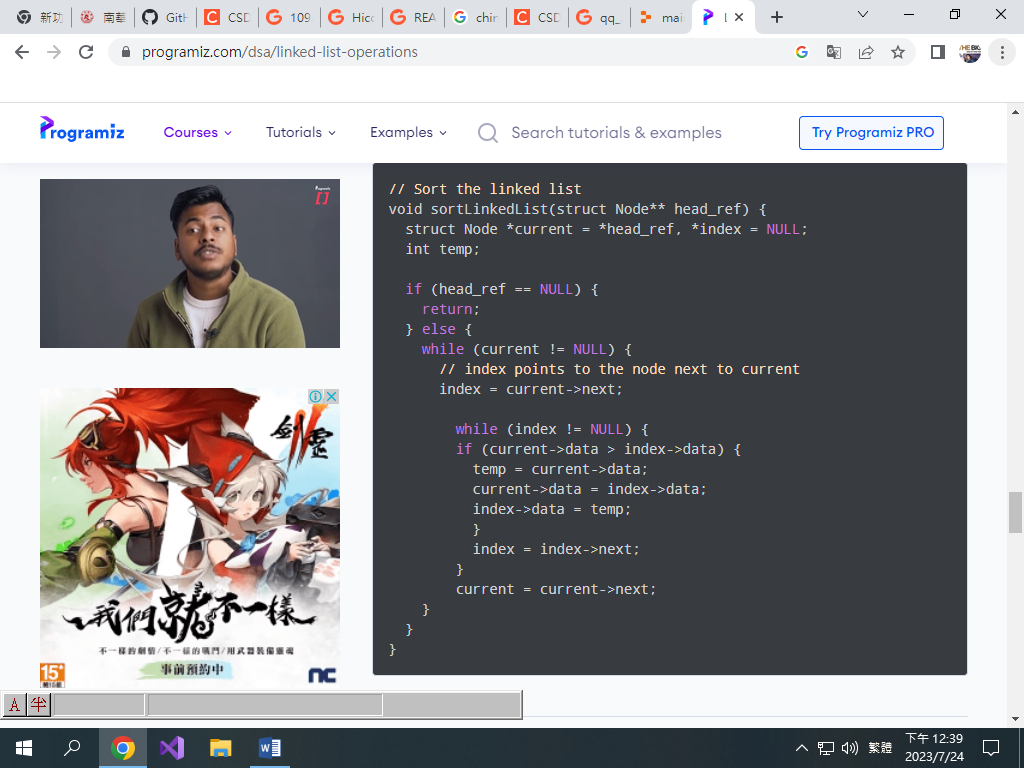


Figure 8: Sort of elements of the Linked list

So we can see the output of the our code (figure 9)

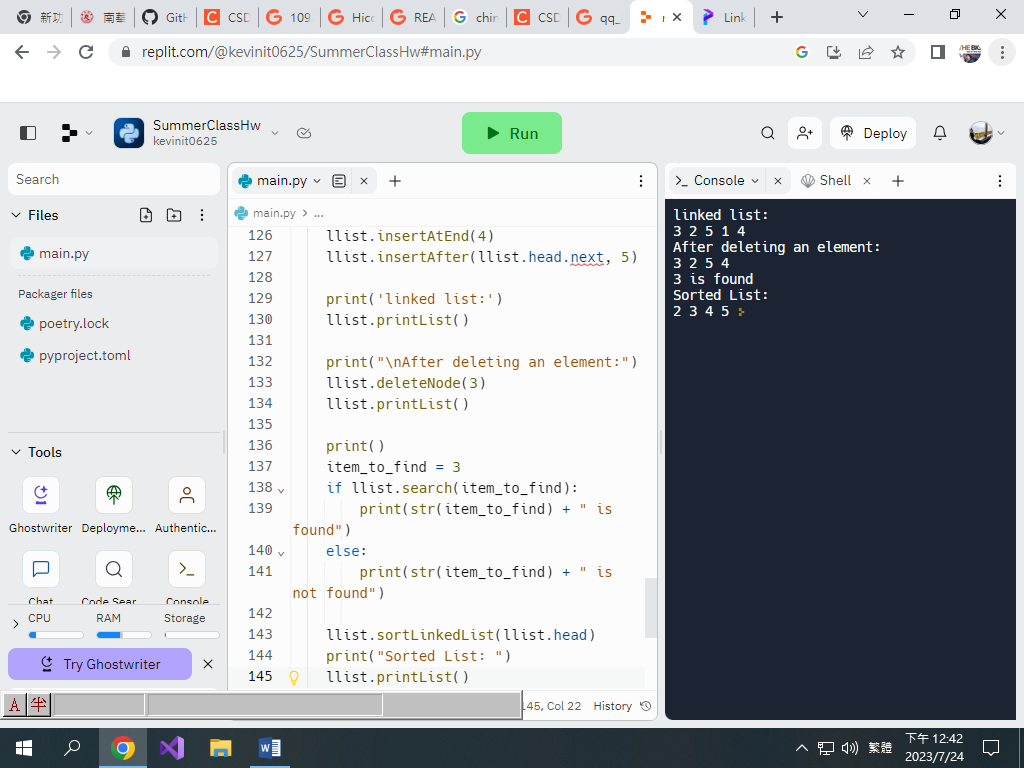


Figure 9: result of the code