Bubble sort: trace 3

Practice 1

Geraint is using the **bubble sort** algorithm to put their list of top five countries to visit into alphabetical order, from A to Z.

Using the `countries` list below, fill in the blanks to show the order of the countries after **each pass of the bubble sort algorithm**, using a new row for each pass. The algorithm will pass over the entire list four times.

`countries`				
Laos	Egypt	Peru	Fiji	Cuba

Items:

Cuba Egypt Fiji Laos Peru

Bubble sort: purpose of temp

Practice 1

The **bubble sort** algorithm works by repeatedly going through a list of items, comparing consecutive pairs of items and swapping the items if they are in the wrong order. On each **pass**, the highest value (or lowest, depending on the order that the items are being sorted into) is moved towards the end of the list. This process is repeated until the list is sorted.

Consider the pseudocode below:

Pseudocode

```
PROCEDURE bubble_sort(items)
    num_items = LEN(items)
    FOR pass_num = 1 TO num_items - 1
        FOR index = 0 TO num_items - 2
            IF (items[index] > items[index + 1]) THEN
                temp = items[index]
                items[index] = items[index + 1]
                items[index + 1] = temp
            ENDIF
        NEXT index
    NEXT pass_num
ENDPROCEDURE
```

Explain the purpose of the `temp` variable in the algorithm.

- `temp` is used to store the total number of items in the list.
- `temp` is used to stop the algorithm when the list is sorted.
- `temp` is used to count the number of comparisons between items.
- `temp` is used to store a copy of one of the values that is being swapped.

Quiz:
STEM SMART Computer Science Week 8

Bubble sort: efficiency 1

Challenge 1

A bubble sort algorithm is written in pseudocode below. Study the pseudocode and then select **two** changes that could be made to improve the efficiency of the algorithm.

Pseudocode

```
PROCEDURE bubble_sort(items)
    num_items = LEN(items)
    FOR pass_num = 1 TO num_items - 1
        FOR index = 0 TO num_items - 2
            IF (items[index] > items[index + 1]) THEN
                temp = items[index]
                items[index] = items[index + 1]
                items[index + 1] = temp
            ENDIF
        NEXT index
    NEXT pass_num
ENDPROCEDURE
```

- [ ] The inner for loop could be changed so that the number of repetitions is reduced by 1 after each pass
- [ ] The outer for loop could be changed to a while loop that stops once no swaps are made during a single pass
- [ ] The variable `temp` is not needed when swapping items in this way and could be removed
- [ ] The inner for loop could be changed to a while loop that only swaps items if they are out of order
- [ ] The outer for loop could be changed so that the number of swaps made is reduced by 1 after each pass

Quiz:
STEM SMART Computer Science Week 8

Bubble sort: fix

Challenge 1

Kofi has written a program to sort an array of data into **descending** order. The algorithm is a version of bubble sort and is shown in pseudocode below.

Pseudocode

```
// Initialise variables
items = [28, 40, 21, 25, 30, 27, 25]
num_items = LEN(items)
temp = 0
pass_number = 1
swapped = False

// Continue while swaps have been made and there are more passes to evaluate
WHILE swapped == True AND pass_number <= num_items - 1
   swapped = False
   FOR index = 0 TO num_items - 2
       // Check if items are out of order
       IF items[index] < items[index + 1]
           // Swap items
           temp = items[index]
           items[index] = items[index + 1]
           items[index + 1] = temp
           swapped = True
        ENDIF
    NEXT index
    pass_number = pass_number + 1
ENDWHILE
```

Part A

There is a problem with the algorithm and it will not sort the data correctly. What is the problem?

- ○ The `items` are being compared in the wrong order.
- ○ The outer loop is running insufficient times.
- ○ The inner loop is running too many times.
- ○ The variable `swapped` has been initialised incorrectly.

Part B

Kofi has corrected the error in the pseudocode.

What is the **total number of swaps** required to sort the array `items` into descending order?

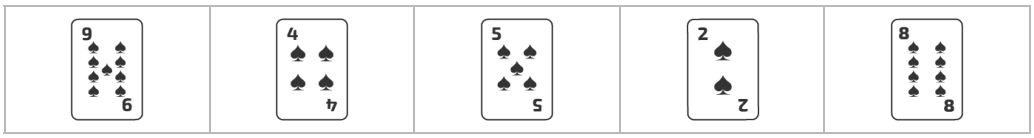
Enter your answer as a **number**.

Quiz:
STEM SMART Computer Science Week 8

Insertion: where to start

Practice 1

You have been given some playing cards that have not been sorted:



In an insertion sort, which would be the first card in the **unsorted** sublist and therefore the **first** card to insert?

- 
- 
- 

Quiz:
STEM SMART Computer Science Week 8

Insertion: trace 2

Practice 1

Geraint is using the **insertion sort** algorithm to put their list of top five countries to visit into alphabetical order, from A to Z.

countries				
Peru	Laos	Fiji	Egypt	Cuba

The insertion sort algorithm has been started on the list below. In the first row, the first item is highlighted to show it is in the sorted sublist and the unsorted sublist contains the remaining items.

In the second row, the first pass has completed and the sorted sublist now contains two items, which are highlighted.

Fill in the blanks to show the order of the countries at the end of each pass when performing an **insertion sort**. Use a new row for each pass.

countries				
Peru	Laos	Fiji	Egypt	Cuba
Laos	Peru	Fiji	Egypt	Cuba

Items:

Cuba Egypt Fiji Laos Peru

Quiz:
STEM SMART Computer Science Week 8

Insertion: complete algorithm

Practice 2

The **insertion sort** algorithm works by building up a sorted sublist, one item at a time. To do that, one by one the items in the list are examined and inserted into the correct position in the sorted sublist. The sorted sublist grows until the whole list is sorted.

The following pseudocode subroutine is an implementation of the insertion sort algorithm. There is a block of code missing.

Drag and drop the statements into the correct order to complete the missing part of the code. Make sure that you **indent** the statements as appropriate.

Pseudocode

```
1 PROCEDURE insertion_sort (items)
2     num_items = LEN(items)
3     FOR index = 1 TO num_items - 1
4         item_to_insert = items[index]
5         previous = index - 1
6         >>>>  [missing block of code]  <<<<
7     NEXT index
8 ENDPROCEDURE
```

Available items

```
items[previous + 1] = items[previous]
```

```
previous = previous - 1
```

```
items[previous + 1] = item_to_insert
```

```
ENDWHILE
```

```
WHILE previous >= 0 AND items[previous] > item_to_insert
```

Quiz:
STEM SMART Computer Science Week 8

Merge sort: trace 1

Practice 1

Amaia coaches a school basketball team. She keeps the scores that the team got in the latest tournament in a list called `basketball_finals`. You can see the items of the list below:

basketball_finals				
250	120	95	101	80

Amaia wants to use the merge sort algorithm to sort the scores from lowest to highest value. Fill in the gaps with the values to show the order of the items after the **first merge**. Assume that the splitting stage has already completed and each value is in a list of its own.

Final split				
250	120	95	101	80
Merge 1				

Items:

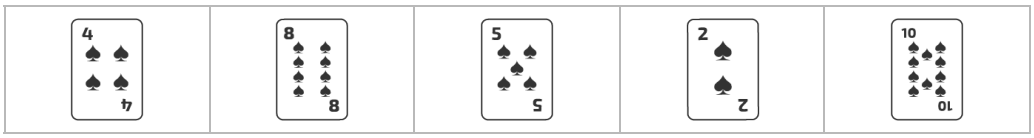
80 95 101 120 250

Quiz:
STEM SMART Computer Science Week 8

Merge sort: trace 2

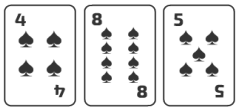
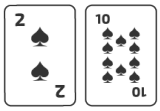
Practice 2

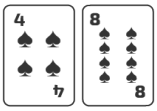
You have been given some playing cards that have not been sorted:



Apply a **merge** sort to sort the cards from the lowest to highest value.

The first three steps of splitting the data have been completed for you:

	Group 1	Group 2
Step 1		

	Group 1	Group 2	Group 3	Group 4
Step 2				

	Group 1	Group 2	Group 3	Group 4	Group 5
Step 3					

Drag and drop the cards provided at the bottom of the question into the correct positions to show the **merge** steps of the algorithm:

	Group 1	Group 2	Group 3
Step 4			

	Group 1	Group 2
Step 5		

	Sorted group
Step 6	

Items:

Quiz:

STEM SMART Computer Science Week 8

Merge sort: trace 6

Practice 2

The following words are stored in an array:

```
Purple, Yellow, Green, Red, Blue, Magenta
```

A merge sort will be used to sort the words into **ascending** alphabetical order so that, when the sorting is complete, the order of the words will be:

```
Blue, Green, Magenta, Purple, Red, Yellow
```

The algorithm uses the recursive method which firstly splits the list into sublists and then merges the sublists.

Which one of these shows the sublists **one step before** the list is completely sorted?

List 1		List 2		List 3	
Purple	Yellow	Green	Red	Blue	Magenta

List 1				List 2	
Green	Purple	Red	Yellow	Blue	Magenta

List 1					List 2
Blue	Green	Purple	Red	Yellow	Magenta

List 1			List 2		
Green	Purple	Yellow	Blue	Magenta	Red