Insertion: complete algorithm

A Level P P Adv Higher P P

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

```
PROCEDURE insertion_sort (items)
    num_items = LEN(items)
    FOR index = 1 TO num_items - 1
        item_to_insert = items[index]
        previous = index - 1
        >>>>  [missing block of code]  <<<<
    NEXT index
ENDPROCEDURE
```

Available items

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

```
previous = previous - 1
```

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

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

```
ENDWHILE
```

Dictionary: trace algorithm

A Level

C C

Abe has been writing a program that generates a festive name.

To create the festive name, the program looks up each letter of the person's name in the `words` dictionary. The corresponding value associated with each letter key is concatenated to the `festive_name` variable without any spaces.

For example, the name `Abe` should generate the festive name `AntlerBellElf`.

The code within the `for` loop is incomplete. What should the missing code be?

Pseudocode

```
DICTIONARY words = {
    "A":"Antler",
    "B":"Bell",
    "C":"Carol",
    "D":"Decorations",
    "E":"Elf"
}

// Ask user for a name and convert it to uppercase
name = INPUT("Enter a name: ")
name = UPPER(name)

festive_name = ""

// Create a festive name from the words dictionary and name
FOR i = 0 TO LEN(name)
    // Use the current letter from name as the key of the words dictionary
    festive_name += >>> MISSING CODE >>>
NEXT i

PRINT("Your festive name is " + festive_name)
```

The code within the `for` loop is incomplete. What should the missing code be?

- words[i]
- words[name]
- words[name[i]]

Quiz:
STEM SMART 2024 Residential Catch Up Quiz 2

Recursion: complete missing code

A Level

P P

Fibonacci numbers are a number sequence that starts: 0, 1, 1, 2, 3, 5, 8, 13, ...

Each Fibonacci number is the **sum of the previous two numbers**. The table below shows the first 10 Fibonacci numbers, from F_0 to F_9:

F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9
0	1	1	2	3	5	8	13	21	34

An incomplete recursive function `fibonacci(n)` has been written below to output the nth Fibonacci number, F_n. For example, running the function with the argument value 7 would return 13 (as the Fibonacci number F_7 is 13).

Hint 2 contains a breakdown of the rules for the Fibonacci sequence if you need help with completing the missing code.

Pseudocode

```
FUNCTION fibonacci(n)
    IF n == 0 THEN
        RETURN [a]
    ELSEIF n == 1 THEN
        RETURN [b]
    ELSE
        RETURN [c]
    ENDIF
ENDFUNCTION
```

Part A Code for [a]

What should replace **[a]**?

Part B Code for [b]

What should replace **[b]**?

Part C Code for [c]

What should replace **[c]**?

Quiz:
STEM SMART 2024 Residential Catch Up Quiz 2

Binary or linear search? 2

GCSE

P P

Identify **three** reasons why you may choose to perform a linear search rather than a binary search on a list of data.

- [ ] The algorithm is simpler to implement
- [ ] The list is very long
- [ ] The list is unsorted
- [ ] The list is very short
- [ ] The list is sorted

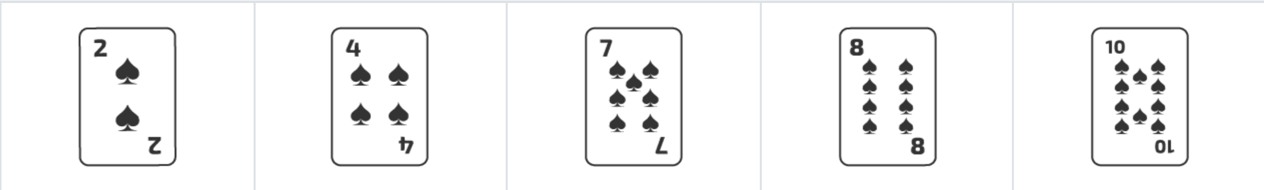
Quiz:
STEM SMART 2024 Residential Catch Up Quiz 2

Binary or linear search? 1

GCSE

C C

You have been provided with some cards that are in the following order:



Part A

You need to find out if the number 4 is in the list of cards. Which searching algorithm would require the **fewest** number of comparisons to find the data?

- Linear
- Binary
- Both would be the same

Part B

You now need to find out if the number 9 is in the list of cards. Which searching algorithm would require the **fewest** number of comparisons to find the data?

- Binary
- Linear
- Both would be the same

Binary search: max comparisons 5

A Level P P Adv Higher P P

You are trying to find a movie in the media library stored on your computer, and decide to use a binary search algorithm to do it.

Your media collection has **100** titles in it. What is the **maximum number of comparisons** that could be made on your media collection while attempting to find a movie?

Here is a pseudocode version of the binary search algorithm:

Pseudocode

```
FUNCTION binary_search(data_set, item_sought)
    index = -1
    found = False
    first = 0
    last = LEN(data_set) - 1
    WHILE first <= last and found == False
        midpoint = (first + last) DIV 2
        IF data_set[midpoint] == item_sought THEN
            index = midpoint
            found = True
        ELSE
            IF data_set[midpoint] < item_sought THEN
                last = midpoint - 1
            ELSE
                first = midpoint + 1
            ENDIF
        ENDIF
    ENDWHILE
    RETURN index
ENDFUNCTION
```

Quiz:
STEM SMART 2024 Residential Catch Up Quiz 2

Merge sort: trace 3

GCSE

P P

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 2024 Residential Catch Up Quiz 2

Bubble sort: efficiency

GCSE C C Adv Higher C C

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 to a while loop that only swaps items if they are out of order
- [ ] 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 outer for loop could be changed so that the number of swaps made is reduced by 1 after each pass
- [ ] The inner for loop could be changed so that the number of repetitions is reduced by 1 after each pass

Quiz:
STEM SMART 2024 Residential Catch Up Quiz 2

Recursion: trace code 5

A Level

C C

A recursive subroutine has been written as follows:

Pseudocode

```
FUNCTION do_something(x, y)
    IF x == 1 THEN
        RETURN y
    ELSE IF y == 1 THEN
        RETURN x
    ELSE
        RETURN do_something(x-1, y-2)
    ENDIF
ENDFUNCTION
```

Trace the subroutine to determine what the final return value will be when the following call is made:

`do_something(4, 8)`

Quiz:
STEM SMART 2024 Residential Catch Up Quiz 2

Bubble sort: complete 2

GCSE C C Adv Higher C C

Put the lines of code provided into the correct order to create a bubble sort algorithm.

Please note that you should use correct indentation in your answer.

Available items

```
items[index + 1] = temp
```

```
END IF
```

```
temp = items[index]
```

```
NEXT index
```

```
items[index] = items[index + 1]
```

```
num_items = LEN(items)
```

```
FOR pass_num = 1 TO num_items - 1
```

```
NEXT pass_num
```

```
IF (items[index] > items[index + 1]) THEN
```

```
FOR index = 0 to num_items - 2
```