Array of records: advantages

Practice 1

Rudi has made a list of gifts that he would like to buy for his family to celebrate the festive break. He has written a program to help him keep track of his ideas. For each gift idea he wants to store:

- Name of the person (the gift is for)
- Description of the gift
- Likely cost
- Purchased (yes/no)

He has decided that an **array of records** is a suitable data structure for his program.

Why has Rudi decided that a record structure is a suitable container for each gift idea?

○ There is no limit to the number of records he can store.

○ The elements of the record (the fields) can be referred to using an index number.

○ It can hold data of different data types.

Dictionary: trace algorithm

Challenge 2

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 dictiona
    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[name]

○ words[name[i]]

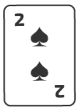
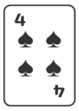
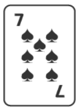
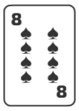
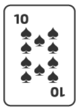
○ words[i]

Binary or linear search? 1

Challenge 1

🟦⬜

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

Quiz:
STEM SMART Computer Science Week 14

Binary search: max comparisons 5

Practice 1

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 Computer Science Week 14

Raspberry Pi Foundation

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 to a while loop that only swaps items if they are out of order
- □ 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 so that the number of swaps made 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

Quiz:
STEM SMART Computer Science Week 14

Bubble sort: complete 2

Challenge 2
🟦🟦

Put the lines of code provided into the correct order to create a bubble sort algorithm. You **must use the correct indentation** in your answer.

Available items

```
NEXT pass_num
```

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

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

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

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

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

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

```
END IF
```

```
NEXT index
```

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

Quiz:
STEM SMART Computer Science Week 14

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 14

Raspberry Pi Foundation

Home STEM SMART Computer Science Week 14 Recursion: complete missing code

Recursion: complete missing code

Practice 2
●●

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 n th 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 Computer Science Week 14

Raspberry Pi Foundation

Recursion: advantages

Challenge 2
🟦🟦

A recursive algorithm can always be written to use iteration (instead of recursion). The following statements show possible advantages of using recursion. Which **two** are correct?

- □ Recursive algorithms use less memory
- □ Problems that are naturally expressed by recursion are easier to code
- □ Recursive algorithms are easier to trace
- □ There are usually fewer lines of code

Quiz:
STEM SMART Computer Science Week 14

Raspberry Pi Foundation

Recursion: trace the code 2

Challenge 2
■■

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)`