Insertion sort algorithm

Task 1 . Demonstrating insertion sort

**Describe** how an insertion sort is performed.

|  |
| --- |

**Show** the steps of an insertion sort on the list of data in **Figure 1** so that the elements are in alphabetical order. Each pass should be on a new line and you should clearly highlight which part of the list is the sorted sublist. The first row has been filled in for you.

| **Element** | Chile | Guyana | Ecuador | Brazil | Peru | Bolivia |
| --- | --- | --- | --- | --- | --- | --- |
| **Index** | 0 | 1 | 2 | 3 | 4 | 5 |

**Figure 1**

| **Chile** | Guyana | Ecuador | Brazil | Peru | Bolivia |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Demonstrate** how an insertion sort would place the following numbers into ascending numerical order:

32, 8, 128, 16, 64, 256

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Task 2 . An insertion sort algorithm

An implementation of an insertion sort in Python is shown in **Figure 2**. Read through the code to familiarise yourself with it - don’t worry if you don’t understand all of it yet.

| 1  2  3  4  5  6  7  8  9 | def insertion\_sort(items):  # Initialise the variables  num\_items = len(items)  # Repeat for each item in the unsorted part of the list  for first\_unordered in range(1, num\_items):  # Copy the first unordered item into value and  # set current to the position before  value = items[first\_unordered]  current = first\_unordered - 1  # Repeat while the start of the list has not been reached  # and the current item is greater than value  while current >= 0 and items[current] > value:  # Copy the item from the current position to the next element  items[current+1] = items[current]  # Proceed to the previous item in the list  current = current - 1  # Copy the value of the first unordered item into the correct position  items[current+1] = value |
| --- | --- |

**Figure 2**

**State** the number of times the outer for loop would repeat if items was a list of 10 items.

**Hint:** the first value of range is the start value and the second value is the stop value.

|  |
| --- |

**Describe** what line 3 does during each iteration of the outer for loop.

|  |
| --- |

**Explain** the purpose of the condition items[current] > value on line 6.

|  |
| --- |

**Complete** the trace table below only for lines 6-9 of the algorithm. The first line in the trace table contains the items list after two passes of the algorithm (first\_unordered in now 3). The variables value and current after executing lines 4 and 5 have also been included in the table.

|  |  |  | items | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Line | value | current | [0] | [1] | [2] | [3] | [4] |
|  |  |  | Abeer | Lola | Yara | Carlos | Tami |
| 4 | Carlos |  |  |  |  |  |  |
| 5 |  | 3 |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

**Explain** the purpose of line 7-8 in the insertion sort algorithm in **Figure 2** using the table above as an example.

|  |
| --- |

What happens when line 9 is omitted from the algorithm in **Figure 2**?

|  |
| --- |