Binary search algorithm

Task . A binary search algorithm

An implementation of a binary search in Python is shown in **Figure 1**. 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  10  11  12  13  14  15 | def binary\_search(items, search\_item):  # Initialise the variables  index = -1  first = 0  last = len(items) - 1  found = False  # Repeat while there are still items between first and last  # and the search item has not been found  while first <= last and found == False:  # Find the middle item (midpoint) between first and last  midpoint = (first + last) // 2  # Compare the item at the midpoint to the search item  if items[midpoint] == search\_item:  index = midpoint  found = True  elif items[midpoint] < search\_item:  first = midpoint + 1 # Focus on right half of range  else:  last = midpoint - 1 # Focus on the left half of range  return index |
| --- | --- |

**Figure 1**

**State** the data type of the variable found in **Figure 1**.

|  |
| --- |

Which of the following statements is **true**?

1. The algorithm in **Figure 1** uses nested iteration.
2. The algorithm in **Figure 1** uses indefinite iteration.
3. The algorithm in **Figure 1** will loop infinitely.
4. The algorithm in **Figure 1** uses nested selection.

**Explain** why the calculation of midpoint in line 9 uses floor division.

|  |
| --- |

**Complete** the trace table below using the algorithm in **Figure 1** when items are in the list [“Ahmed”, “Chloe”, “Keira”, “Olivia”, “Neelu”, “Reg”, “Steph”, “Zak”] and search\_item is “Neelu”.

The first pass has been completed for you.

| Line | index | first | last | found | midpoint | items[midpoint] | **Condition** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | -1 |  |  |  |  |  |  |
| 3 |  | 0 |  |  |  |  |  |
| 4 |  |  | 7 |  |  |  |  |
| 5 |  |  |  | False |  |  |  |
| 6 |  |  |  |  |  |  | True |
| 7 |  |  |  |  | 3 |  |  |
| 8 |  |  |  |  |  | Olivia | False |
| 11 |  |  |  |  |  | Olivia | True |
| 12 |  | 4 |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |