**Higher Computing**

**Dry run tasks**

**Task 1**

1. SET total TO 0
2. FOR counter FROM 1 TO 5 DO
3. RECEIVE userInput FROM (INTEGER)KEYBOARD
4. SET total TO total + userInput
5. END FOR
6. SET average TO total / 5
7. SEND "The average of these numbers was "& average TO DISPLAY

|  |  |  |  |
| --- | --- | --- | --- |
| **counter** | **userInput** | **total** | **average** |
| 1 | 3 |  |  |
| 2 | 7 |  |  |
| 3 | 4 |  |  |
| 4 | 11 |  |  |
| 5 | 11 |  |  |

**Task 2**

Consider the following VB code:

For n = 1 To 5

 If rating(n) = 1 Then

 onestar = onestar + 1

 ElseIf rating(n) = 2 Then

 twostar = twostar + 1

 ElseIf rating(n) = 3 Then

 threestar = threestar + 1

 End If

Next n

Copy and complete the trace table for the end of each loop, when the array rating is stored as follows:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Index** | **1** | **2** | **3** | **4** | **5** |
| **Rating** | 3 | 3 | 2 | 1 | 2 |

|  |  |  |  |
| --- | --- | --- | --- |
| **n** | **Onestar** | **Twostar** | **Threestar** |
| **1** |  |  |  |
| **2** |  |  |  |
| **3** |  |  |  |
| **4** |  |  |  |
| **5** |  |  |  |

**Task 3**

Consider the following algorithm.

|  |
| --- |
| SET highest to marks(1)Set position to 0FOR index = 2 to 5  IF marks(index) > highest THEN Highest = marks(index) Position = index END IFNEXT indexSEND pupils(position) to DISPLAY |

1. Describe the purpose of the code above.

|  |
| --- |
|  |

1. Complete the trace table below to show the value of variables if the program contained the following data.

|  |  |  |
| --- | --- | --- |
| **Index** | **Highest** | **Position**  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |  |
| --- | --- | --- |
|  | Pupil() | Marks() |
| 1 | Sam | 25 |
| 2 | Paula | 28 |
| 3 | Daisy | 24 |
| 4 | Steph | 29 |
| 5 | Jason | 29 |

1. Describe the problem with the code above.

|  |
| --- |
|  |

**Task 4**

|  |  |  |  |
| --- | --- | --- | --- |
| **9** | **6** | **10** | **9** |

1. PROCEDURE linearSearch(numbers,)
2. SET itemToFind TO 10
3. SET found TO false
4. SET arraySize TO 4
5. SET counter TO 0
6. REPEAT
7. IF number[counter] = itemToFind THEN
8. SET found TO true
9. END IF
10. SET counter TO counter + 1
11. UNTIL found OR counter > arraySize
12. IF found THEN
13. SEND itemToFind &" found at position" & counter - 1 TO DISPLAY
14. ELSE
15. SEND "Item not found" TO DISPLAY
16. END IF
17.
18. END PROCEDURE

|  |  |  |  |
| --- | --- | --- | --- |
| **itemToFind** | **found** | **arraysize** | **counter** |
| 10 |  |  |  |
| 10 |  |  |  |
| 10 |  |  |  |
| 10 |  |  |  |

**Task 5**

SET numbers TO [3, 15, 4, 7, 8]

PROCEDURE findMaximum(numbers)

 SET maximumValue TO numbers[0]

 FOR counter FROM 1 TO 4 DO

 IF maximumValue < numbers[counter] THEN

 SET maximumValue TO numbers[counter]

 END IF

 END FOR

 SEND "The largest value was "& maximumValue TO DISPLAY

END PROCEDURE

|  |  |
| --- | --- |
| **Counter** | **maximumValue** |
|  |  |
|  |  |
|  |  |
|  |  |

**Task 6 (Extension)**

Consider the following binary search algorithm:

|  |  |
| --- | --- |
| **1** | Set *found* = FALSE |
| **2** | Set *first\_location* = 0 |
| **3** | Set *last\_location* = length of list - 1 |
| **4** | Get target item from user |
| **5** | START LOOP |
| **6** |  Set *middle* = Int ((*first\_location* + *last\_location*) / 2) |
| **7** |  IF list (*middle*) = target item THEN |
| **8** |  *found* = TRUE |
| **9** |  Display “Found” message |
| **10** |  ELSE IF list (*middle*) < target item THEN |
| **11** |  *first\_location* = *middle* + 1 |
| **12** |  ELSE |
| **13** |  *last\_location* = middle -1 |
| **14** | END LOOP – UNTIL *found* = TRUE OR *first\_location* = *last\_location* |

Copy and complete the trace table for when the list is stored as follows, and **target item = 37**:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Index** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** |
| **List** | 21 | 37 | 39 | 68 | 99 | 102 | 109 | 154 | 188 | 349 | 471 | 479 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **End of pass** | **Found** | **First\_location** | **Last\_location** | **Middle** |
| **Initial state** |  |  |  |  |
| **1** |  |  |  |  |
| **2** |  |  |  |  |
| **3** |  |  |  |  |

Loop ends because first\_location = last\_location.