Tracing code

Task 1 . Russian multiplication

The Python code in **Figure 1** is an implementation of the Russian multiplication algorithm. This method calculates the product of two numbers as a sum by using **integer division** and **modulo (MOD)**. Use the table below to help you investigate the algorithm in Python.

|  | **Explanation** | **Python example** |
| --- | --- | --- |
| **Modulo (MOD)** | Calculates the **remainder** of a division. For example 7 MOD 3 will calculate as 1. | 7 % 3 |
| **Integer division** | Calculates the **whole** number of times the divisor (3) will go into the dividend (7). For example 7 ÷ 3 will calculate as 2. | 7 // 3 |

| 1  2  3  4  5  6  7  8  9  10 | print("Numbers:")  a = int(input())  b = int(input())  sum = 0  while b > 0:  if b % 2 == 1:  sum = sum + a  a = 2\*a  b = b // 2  print(sum) |
| --- | --- |

**Figure 1**

**State** the result of the following calculation in Python: 14 % 4

|  |
| --- |

**State** the result of the following calculation in Python: 28 // 5

|  |
| --- |

**Complete** the trace table below using the algorithm in **Figure 1**. The values of a and b have been provided and the first iteration of the while loop has been filled in for you.

| Line | a | b | sum | **Condition** | **Output** |
| --- | --- | --- | --- | --- | --- |
| 1 |  |  |  |  | "Numbers:" |
| 2 | 11 |  |  |  |  |
| 3 |  | 7 |  |  |  |
| 4 |  |  | 0 |  |  |
| 5 |  |  |  | True |  |
| 6 |  |  |  | True |  |
| 7 |  |  | 11 |  |  |
| 8 | 22 |  |  |  |  |
| 9 |  | 3 |  |  |  |
| 5 |  |  |  | True |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Explain** whether the algorithm in **Figure 1** will loop infinitely or not.

|  |
| --- |

Task 2 . Lowest number in a list

In this task, you are going to analyse a piece of code to check if it is working correctly. The program is meant to find the lowest number from a list of integers called items and store the lowest value from this list in the variable lowest.

| 1  2  3  4 | lowest = items[0]  for current in range(1, len(items)):  if lowest < items[current]:  lowest = items[current] |
| --- | --- |

**Figure 2**

**Complete** the trace table below using the algorithm in **Figure 2**. The list of items and the first two lines of code have been filled in for you.

|  |  |  |  | items | | | | |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Line | lowest | current | **Condition** | [0] | [1] | [2] | [3] | [4] |
|  |  |  |  | 24 | 16 | 35 | 42 | 7 |
| 1 | 24 |  |  |  |  |  |  |  |
| 2 |  | 1 |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

**Explain** whether the algorithm in **Figure 2** works as intended.

|  |
| --- |

Task 3 . Nested loops

The algorithm in **Figure 3** contains a nested loop; a loop within a loop. The outer for loop has a lower number of iterations then the inner loop. **Note** that the end number of the range is not included in the generated sequence because it is used as the stop point.

| 1  2  3  4  5 | total = 0  for i in range(1,3):  for j in range(2,5):  total = total + j  print(total) |
| --- | --- |

**Figure 3**

**Complete** the trace table below using the algorithm in **Figure 3**.

| Line | total | i | j | **Output** |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |