5 A global 1D array `Num` of integers contains four elements. A program assigns values to these elements as shown:

```
Num[1] ← 1
Num[2] ← 2
Num[3] ← 5
Num[4] ← 3
```

A procedure `Process()` manipulates the values in the array.

The procedure is written in pseudocode:

```
PROCEDURE Process(Start : INTEGER)
   DECLARE CaseVar, Index, Count : INTEGER

   Index ← Start
   Count ← 0

   WHILE Count <= 20
      CaseVar ← Num[Index]
      CASE OF CaseVar
         1 : Num[Index] ← Num[Index] + Index //clause one
             Index ← Index + 1
             Count ← Count + 1

         2 : Num[Index] ← Num[Index] + Index //clause two
             Index ← Index + 2
             Count ← Count + 2

         3 : Num[Index] ← Num[Index] * 2     //clause three
             Index ← Index + 3
             Count ← Count - 1

         4 : Count ← Count + 4               //clause four
         OTHERWISE : Count ← 20
      ENDCASE

      Index ← (Index MOD 4) + 1
   ENDWHILE

ENDPROCEDURE
```

(a) Complete the trace table by dry running the procedure when it is called in the statement:
`CALL Process(1)`

Index	CaseVar	Count	Num[1]	Num[2]	Num[3]	Num[4]

[5]

(b) As a reminder, the `CASE` structure in the pseudocode is:

```
CASE OF CaseVar
   1 : Num[Index] ← Num[Index] + Index //clause one
       Index ← Index + 1
       Count ← Count + 1

   2 : Num[Index] ← Num[Index] + Index //clause two
       Index ← Index + 2
       Count ← Count + 2

   3 : Num[Index] ← Num[Index] * 2     //clause three
       Index ← Index + 3
       Count ← Count - 1

   4 : Count ← Count + 4               //clause four
   OTHERWISE : Count ← 20
ENDCASE
```

The `CASE` structure could be optimised by combining two existing clauses.

(i) Identify the `CaseVar` values for these **two** existing clauses.

..... [1]

(ii) Write pseudocode for a single clause to replace the **two** clauses identified in **(b) (i)**.

.....

.....

..... [2]