Topic: 2.1.3 Corrective maintenance
2.1.4 Adaptive maintenance

Trace table - a technique used to test algorithms to make sure that no *logical* errors occur.

Hand tracing or 'dry running' allows you to use a **trace table** to

- see what code will do before you have to run it
- locate where errors in your code are

Taking a program like the one below we need to keep track (trace) all the variables and outputs.

```
Dim y as integer = 3
For x = 1 to 4
       y = y + x
Loop
Console.writeline(y)
```

To do this we create a trace table:

x	y	output
1	3	
2	4	
3	6	
4	9	
4	13	13

The exam will normally ask you to create a trace table of some sort so you need to be very confident with them. The exam will usually give you the headings but just in case; there are several steps in making a trace table,

Topic: 2.1.3 Corrective maintenance
2.1.4 Adaptive maintenance

The first one is to note the table headings, this involves the following:

1. **VARIABLES: note all the variables in the piece of code you are looking at (this includes arrays). Note each variable as a heading**
2. **OUTPUTS: note if there is an output and put this as a heading**
3. **INPUTS: if there are inputs specified, put an inputs column and be prepared to fill it in.**

It is very easy to jump right in when filling in trace tables, but you must be careful. The exam will try and trick you, so trying to predict what a trace table will do isn't a good idea. In fact, the best idea is to switch your brain off and tackle the problem line by line, exactly as a computer would. Take a look at the following example:

Example: Simple trace table

```
Dim num() as integer = {10,8,3,5,6,1,2}
Dim sum as integer = 0
Dim avg as decimal
For x = 0 to 5
        sum = sum + num(x)
Loop
avg = sum / (x + 1)
Console.writeline("average =" & avg)
```

1. note all the variables: num array / sum / avg / x
2. note if there is an output: yes
3. if there are inputs specified: no

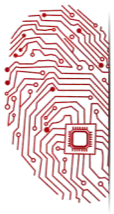
So we should construct the following table:

num										
0	**1**	**2**	**3**	**4**	**5**	**6**	**sum**	**avg**	**x**	**output**
10	8	3	5	6	1	2	0			
								0		

Now looking at the names of the variables you might be tempted to add all the values in the array together to find the sum, and then find the average number from this calculation "35/7 =5". However, you'd be wrong, create a trace table and see if you can find the correct answer:

Answer :

num										
0	**1**	**2**	**3**	**4**	**5**	**6**	**sum**	**avg**	**x**	**output**
10	8	3	5	6	1	2	0			
							10		0	
							18		1	
							21		2	
							26		3	
							32		4	
							33	5.5	5	average =5.5



Topic: 2.1.3 Corrective maintenance
2.1.4 Adaptive maintenance

So what went wrong? If you look at the trace table you can see that we never added the number 2 from the num array to the sum, it stopped at element 5. To fix this we would adjust the following line:

"For x = 0 to 6"

Complete the trace table for the following code:

```
Dim nums( )= {6,2,8,1,9,2}
Dim n as Integer = 0

For i = 0 to 5
If nums(i)> n
    n = nums(i)
endif
loop
```

Answer:

i	n	nums					
		0	1	2	3	4	5
	0	6	2	8	1	9	2
0	6						
1							
2	8						
3							
4	9						
5							

What function does the above code perform?

Answer:

It finds the highest value in an array of values.