Trace Tables - Reading

It is useful when writing programs to be able to make a **dry run** of the pseudocode or the program that has been produced. **Dry runs** are carried out by programmers rather than computers. To perform a dry run the following process is carried out:

1. For each input choose appropriate **test data** for the algorithm
2. Execute the **algorithm** line by line on paper until it is completed
3. If the program gives the intended result then it will work for that test case
4. If the program failed to give the correct result then fix the **program logic**.

As the program is executed we make use of a **trace table** to write down the **value** of each **variable** as it changes. We can also record the **line number** of the **instruction** that was executed and any **output** from the algorithm.

Algorithm

```
1  timesTable ← 5
2  FOR i ← 1 TO 3
3      result ← i * 5
4      OUTPUT result
5  ENDFOR
```

This algorithm calculates and outputs the first three values for the five times table. The trace table for the algorithm is shown to the right.

Trace table

Line	timesTable	i	result	OUTPUT
1	5			
2		1		
3			5	
4				5
2		2		
3			10	
4				10
2		3		
3			15	
4				15
5				

It is often difficult to **debug** a program that runs but has **logical errors** meaning it doesn't do what you want it to. The advantage of using dry runs and trace tables is that you can carefully find where logical errors and **bugs** in your program are. This is especially useful if you are finding programming difficult or have just started to learn how to program.

Algorithm

```
1  age ← 15
2  price ← 10
3  ticket ← ''
4  IF age ≥ 16 THEN
5    ticket ← 'Adult'
6  ELSE
7    ticket ← 'Child'
8    price ← price * 0.5
9  ENDIF
10 OUTPUT Ticket
11 OUTPUT price
```

Trace table

Line	age	price	ticket	OUTPUT
1	15			
2		10		
3			''	
4				
6				
7			'Child'	
8		5		
9				
10				Child
11				5

An algorithm and trace table to calculate a half price ticket for a child are shown above.