Sorting 1 – Bubble Sort - Reading

We often want to **sort** data stored in computers. This might be surnames of students in a school or house prices in a town. There are many **algorithms** which we can use to sort data. Three well-known ones are **Bubble Sort**, **Insertion Sort** and **Merge Sort**.

Each of the algorithms differs in how quickly it can sort data. If the algorithm sorts the data quickly then we say that it is **efficient**.

This reading looks at the Bubble Sort algorithm. The algorithm works as follows:

- Start with the first number and compare it to the second
- If the numbers are not in the correct order then swap them over
- Now continue to go through each of the numbers comparing and swapping if necessary
- When we reach the end of numbers we have completed the first pass
- Repeat this process with new passes until no pair needs to be swapped. This final pass shows that the list has been correctly ordered.

Below is an example of how to sort the following numbers: 5 1 3 2 4 7 6

FIRST PASS

List of numbers to be sorted	Comments
5 1 3 2 4 7 6	Compare 5 and 1. 5 > 1 therefore swap them
1 **5 3** 2 4 7 6	Compare 5 and 3. 5 > 3 therefore swap them
1 3 **5 2** 4 7 6	Compare 5 and 2. 5 > 2 therefore swap them
1 3 2 **5 4** 7 6	Compare 5 and 4. 5 > 4 therefore swap them
1 3 2 4 **5 7** 6	Compare 5 and 7. 5 < 7 therefore leave them
1 3 2 4 5 **6 7**	Compare 7 and 6. 7 > 6 therefore swap them

SECOND PASS

List of numbers to be sorted	Comments
1 3 2 4 5 6 7	Compare 1 and 3. 1 < 3 so leave them
1 **3 2** 4 5 6 7	Compare 3 and 2. 3 > 2 therefore swap them
1 2 **3 4** 5 6 7	Compare 3 and 4. 3 < 4 so leave them
1 2 3 **4 5** 6 7	Compare 4 and 5. 4 < 5 so leave them
1 2 3 4 **5 6** 7	Compare 5 and 6. 5 < 6 so leave them
1 2 3 4 5 **6 7**	Compare 6 and 7. 6 < 7 so leave them

THIRD PASS

In the third pass each of the numbers is compared just as with the first two passes. As the numbers are now in order, all of them are left where they are, confirming that the list is correctly sorted. The algorithm would need to make use of a **flag** to store whether any swaps had been needed in the pass. A flag is simply a **variable** storing a **Boolean** (true or false) as to whether swaps were made.

One feature of Bubble Sort is that if a large number is on the left then in one pass it can quickly be moved to the right. We saw this in the first pass with the 5 moving quickly to the right. If, though, a small number is on the right then it will move very slowly to the left. This makes the algorithm inefficient. In fact, it is less efficient than the Insertion Sort and Merge Sort algorithms.