**Your name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Tutor Group:\_\_\_\_\_\_\_\_\_\_**

**How to be a Programmer**

**Introduction**

When programming, the problem to be solved is always described in a ***specification***, e.g.

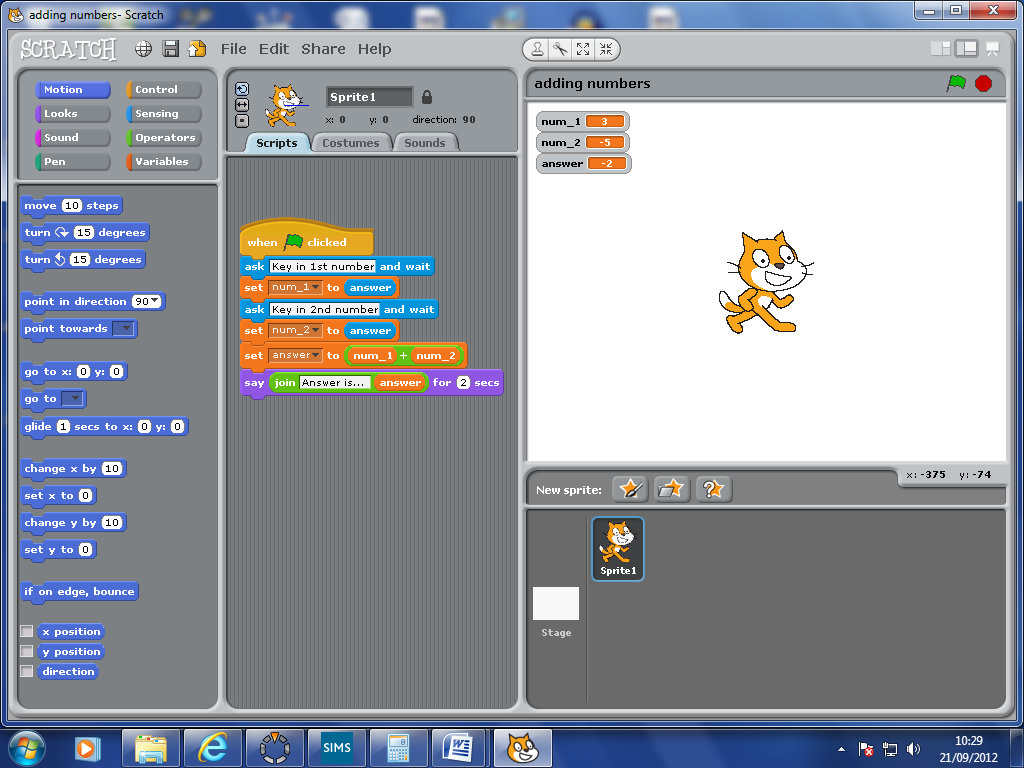
**“Write a programme that inputs two numbers, adds them together, and then outputs the answer”**

Before writing your programme, always begin by jotting down an ***algorithm*** (on paper or use Word). An algorithm is a step by step description of how to solve a problem.

The algorithm for this problem could be:

1. Request first number
2. Input first number and store in variable ***num\_1***
3. Request second number
4. Input second number and store in variable ***num\_2***
5. Perform calculation (***answer*** = ***num\_1*** + ***num\_2***)
6. Output answer

Once you’ve got an algorithm, next think about the ***variables*** (storage locations) you will need. We need three here: one for each of the two numbers and one for the answer. Then try to convert it into code.



Output statement

Calculation

Statements which prompt for input

Variables

Programmes should always be tested to make sure they work. This programme should be tested with mixtures of positive and negative integers (whole numbers) and decimal numbers.

**TASK 1: Create this programme in Scratch and then test it**

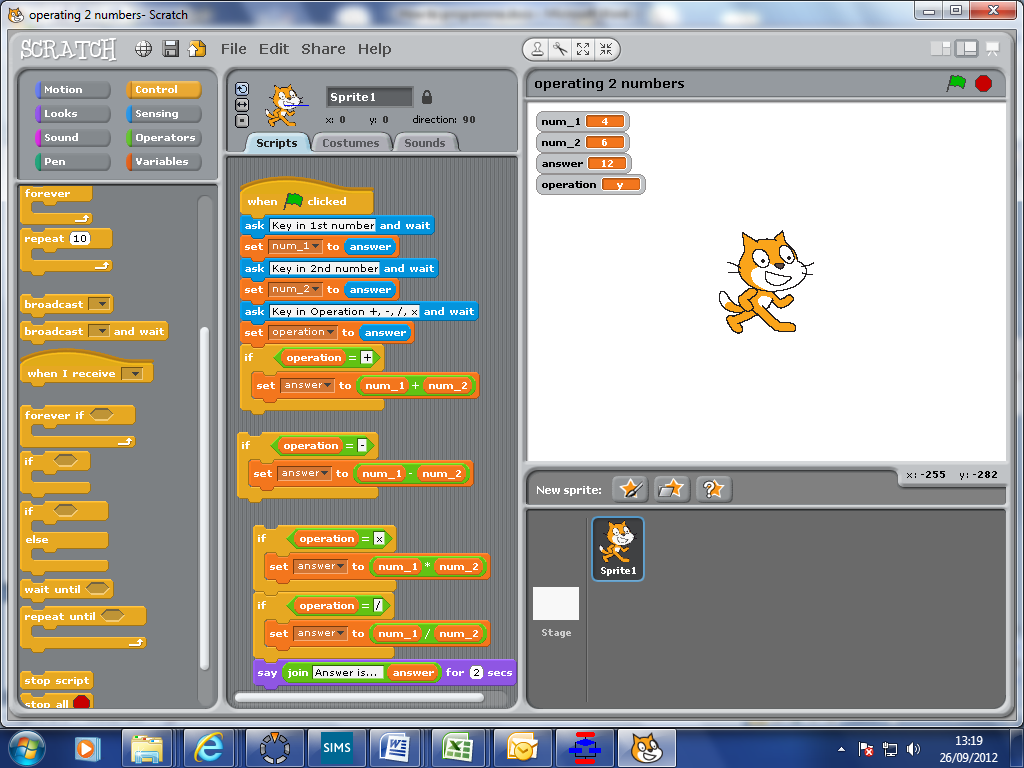
**TASK 2: Plan for this harder specification by writing an algorithm (use Word or paper):**

**“Write a programme that inputs two numbers, and then will either add them, multiply them, divide them or subtract them depending on what the user wants. The programme then outputs the answer”**

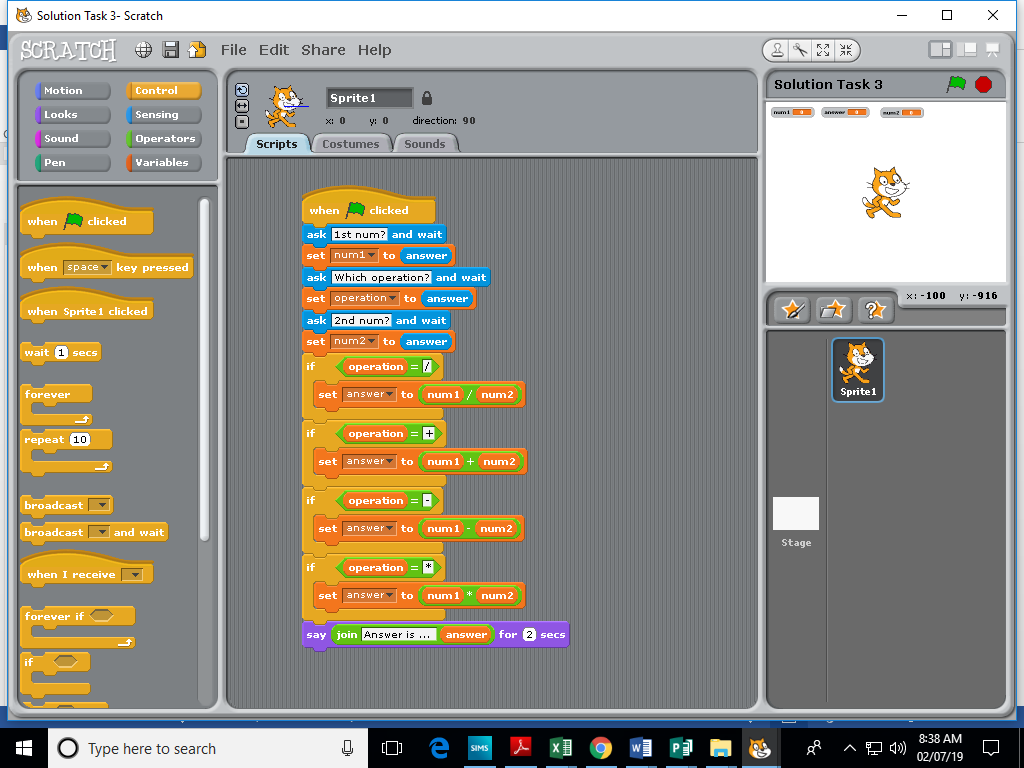
**IF statements**

Thinking about this new specification makes us realise that we need a way of making decisions, i.e. which operation (+, -, x or /) are we to perform on our numbers? We can make decisions using IF statements.

This part of an IF statement is called the ***condition***. **ONLY** if the condition is true are the statements inside the IF executed (carried out).

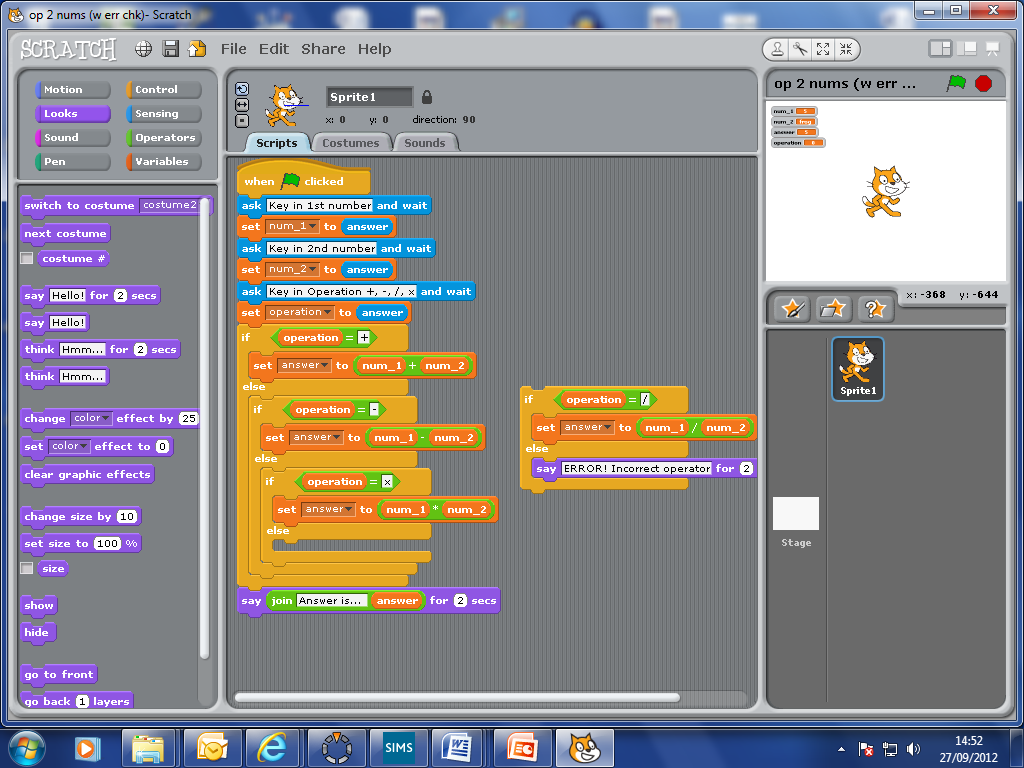


So, we can build the programme along the lines shown here:



As well as IF statements, programming languages also have IF…then…ELSE statements.

The condition is either True or False

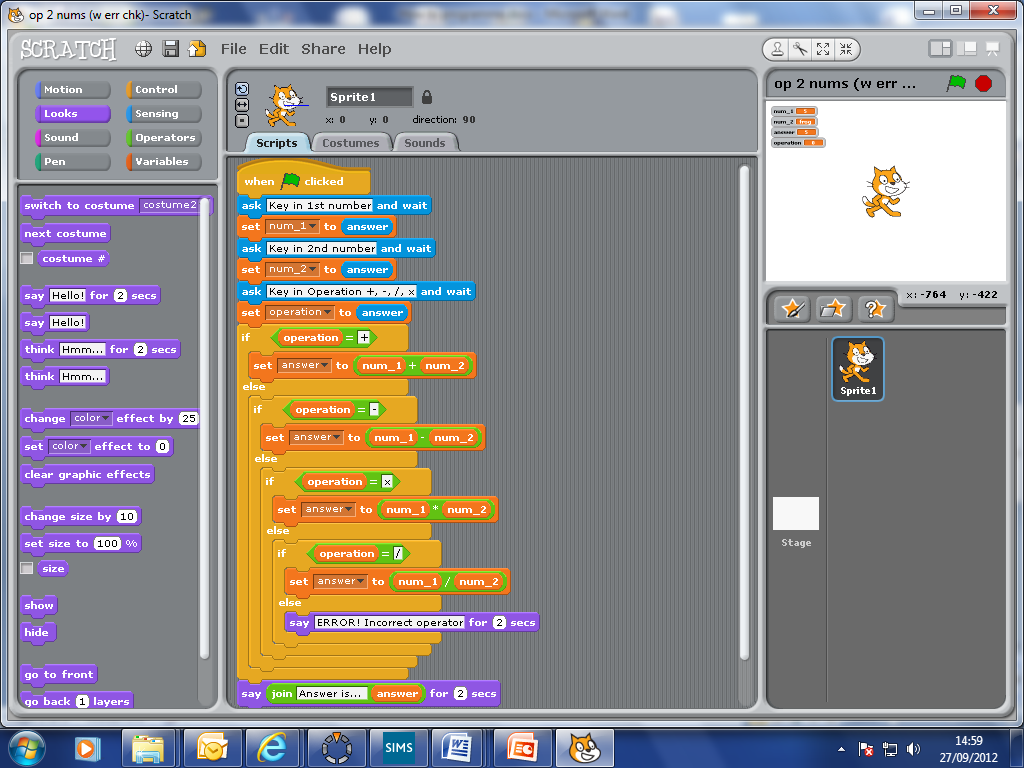


Executed, if condition is false

Executed, if condition is true

Again, the ***condition*** is evaluated to see if it is true or false. If true, the next set of instructions is executed, if false then the programme jumps to the statements after Else.

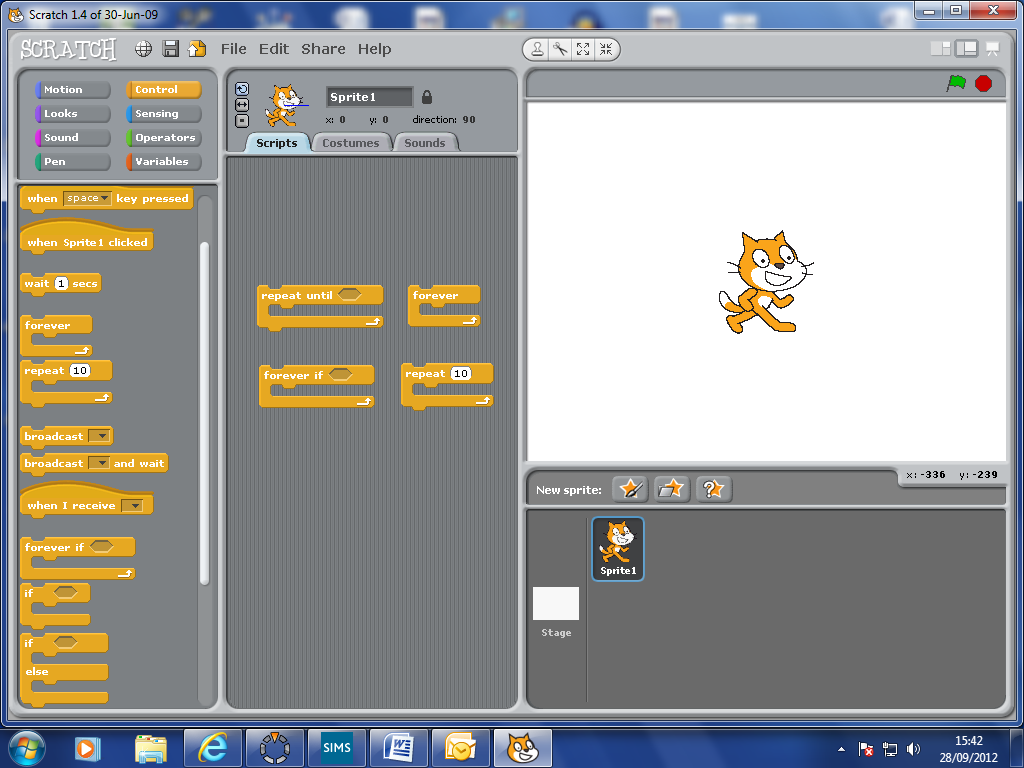
IF statements can also be placed inside each other to form ***nested Ifs***, so an alternative structure for our programme is:



**TASK 3: Now make the programme for this specification (use the pictures above to help you)**

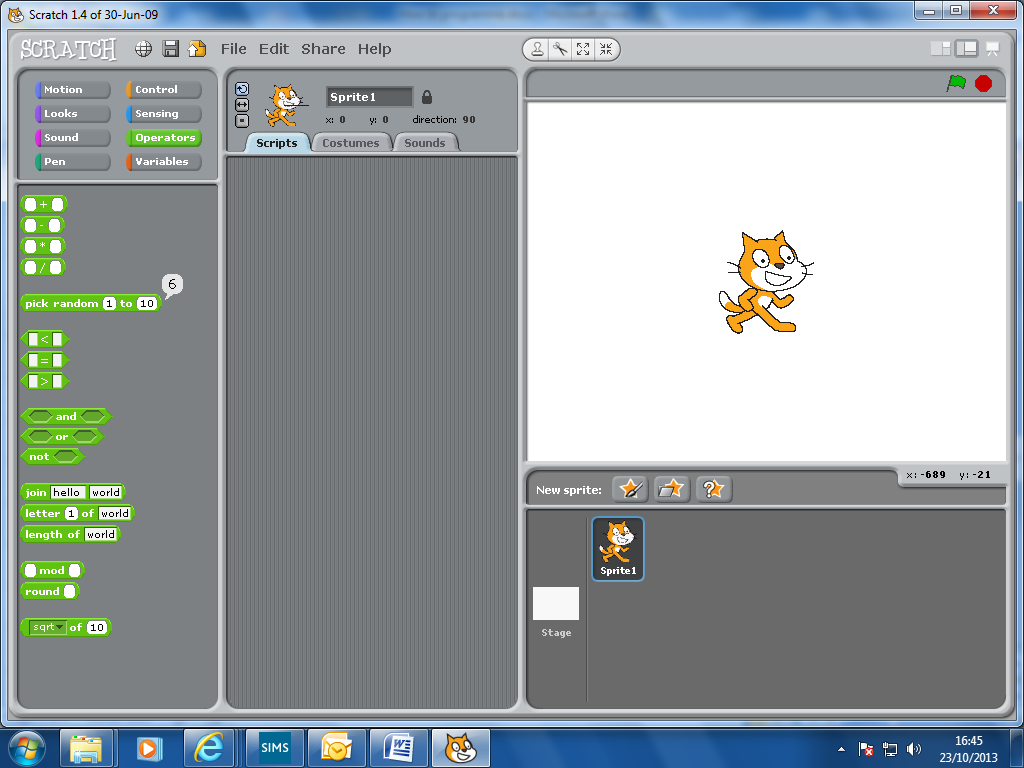
**Loops**

Often in programmes we need to repeat certain sets of instructions. Maybe the instructions need repeating a set number of times, or maybe they need repeating until something else happens (or doesn’t happen!). Loops allow us to do this.



There are four kinds of loop in Scratch. In Python (another programming language) there are just two.

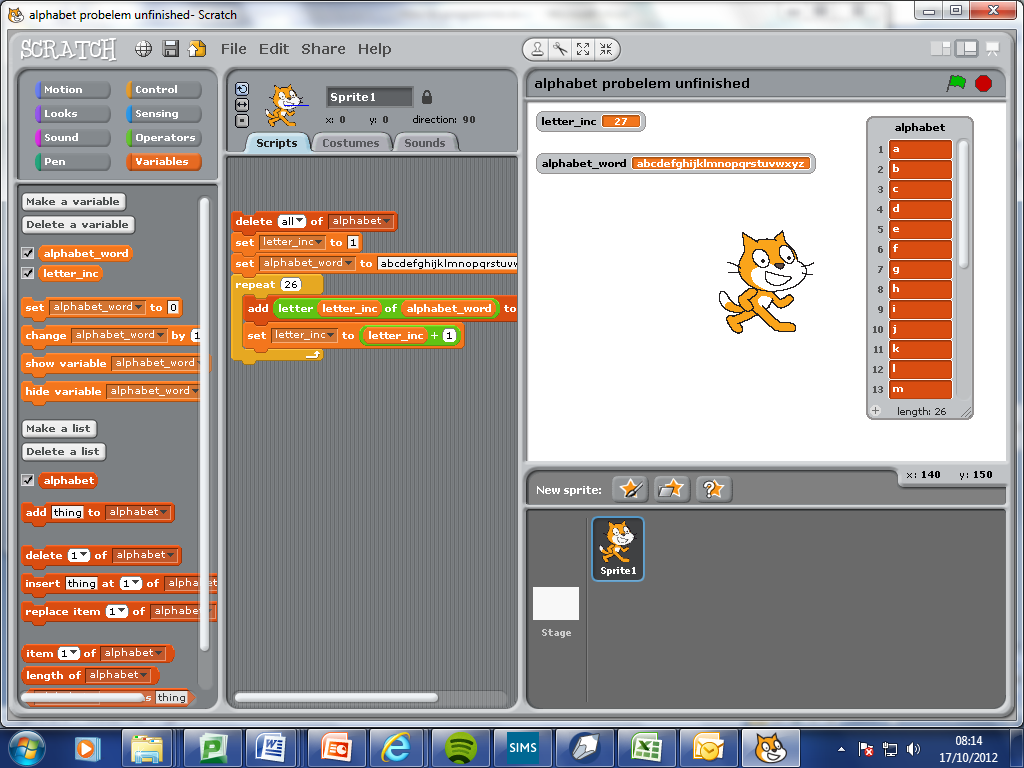
**TASK 4: Write programmes with loops to do each of the following specifications (remember an algorithm will help!):**

1. **Make the cat count up (slowly) from 0 to 10.**
2. **Make the cat count up (slowly) from one user specified number to another (e.g. from 23 to 37 say). Get it to keep a running total of all the numbers as it counts. Display the total once the cat has finished counting.**
3. **Get the cat to say its times-tables. Give the cat the particular number to begin with (e.g. 5 for 5 times table, 7 for 7 times table), it should then count up in multiples. Think about how you will get the cat to stop (that’s up to you). BONUS – make the cat actually say “1 x 5 is 5”, “2 x 5 is 10”, and so on.**
4. **Get the cat to ask for a number followed by a percentage, e.g. 54, 12%. The cat will then calculate what 12% of 54 is, and tell you. The cat will keep asking for more numbers and percentages until you tell it to stop.**
5. **Write a programme to simulate the flipping of a coin a certain number of times. At the end it will tell you the number of heads and tails. (Hint: you will need to use )**

**Lists**

The simplest way to allow a programme to store data is to use variables. Variables are the simplest ***data structures***. They can be pictured as empty boxes to store data in.

However, Scratch also provides ***lists***, which are more complex data structures. A list is a collection of related variables. The strength of a list lies in its ability to allow us to manipulate all members in a specific way. E.g. It is much easier to double each item a twenty item list of numbers than it would be to double each of twenty separately stored numbers.



The items (or ***members***, or ***elements***) of a list are indexed.

A list has a length.

A list has a name.

In most other programming languages, lists are usually called ***arrays***, though in Python they are also known as lists! Commonly, arrays can be multi-dimensional. So a 2D array is very similar to a spreadsheet in structure, i.e. a grid with rows and columns; each row and column having a separate index. In Scratch, lists can only be 1D.

**TASK 5: Now write programmes using lists to do each of the following:**

1. **Write a programme that inputs a word, and then will output the word spelled backwards**
2. **Write a programme that creates a list containing the letters of the alphabet. The cat then asks for a number and a position (START or END). The programme will then output that number of letters either from the start of the alphabet, or that number of letters from the end. E.g 7, END would produce t, u, v, w, x, y, z.**
3. **Write a programme that inputs a series of numbers from the user and stores them in a list. After the last number has been entered the programme tells you which number was largest, which was smallest and what the total is.**

**Harder**

1. **Write a programme that starts with a list of 5 types of fruit. The user is asked to key in a type of fruit to *search* for. If that fruit is in the list the programme outputs where that fruit is in the list, e.g. position 3. If the fruit is not in the list the programme outputs that this fruit has not been found, and then offers to add the fruit to the list.**
2. **Write a programme that will input 5 whole numbers (integers) and store them in a list. The programme will then move the smallest number of the five to the top of the list.**

**How could this programme be improved so that the list of 5 numbers is *sorted* into order (smallest to biggest)? See what you can find out about bubble sorts.**

1. **Write a similar programme to simulate rolling two dice a certain number of times. On each roll the total score of the two dice should be calculated and stored. At the end it will display the number of times each total was reached.**
2. **In a game of Bridge, each of 4 players is dealt 13 cards from a deck. Simulate the dealing of these hands, and show what the contents of each of the players hands is.**