Algorithms & Computational Thinking

Orb education

Contents

1 Computational Thinking

Computers don't think; they only do exactly what you (and the programmers) tell them to do. No more, no less. So computational thinking is not about thinking like a computer. It's about thinking in a way that helps you solve complicated problems.

Aim: *To introduce the ideas of decomposition, pattern recognition, abstraction and algorithms.*

Task 1 – A Day at the Fair

Think about taking a trip to a nearby town or city with your friends to visit a fair or festival. You can decide where the town or city is, but it should be somewhere you can get to on public transport. It's going to be a long day out.

Write down four of the major issues you will have to think about when organising your day out. Don't try and provide answers at this point; just ask the questions. We have started things off for you.

a. *How will we travel to the fair and then back home again?*

b. ____________________

c. ____________________

d. ____________________

You may have solved some of these problems previously. For example, you may have visited the town before so you already know how to get there. If you can, describe something you have done before that will be useful for this trip.

____

____

Write down a few details that are so small or obvious that you don't have to think about them in advance e.g. what seat on the bus you will sit on.

____

____

Now write down some brief instructions for any one of the problems. For example, you may describe your journey.

____

____

____

That's it. You're already well on your way to computational thinking.

A Recipe for Success

An algorithm is a set of instructions which can be followed. Like a recipe in a cookbook, these instructions have to be accurate and in the correct order for the finished product to be as intended.

Aim: *To think about algorithms and branching (selection).*

Task 1 – Making a Sandwich

Place the following instructions in the correct order so that you end up with a nice sandwich rather than a messy pile of ingredients.

a. Place your favourite fillings on one piece of buttered bread.

b. Butter the top side of each piece of bread.

c. Place the other piece of bread upside down over the fillings.

d. Position two slices of bread separately on a plate.

Task 2 – Like a Hot Drink?

Write a set of instructions for making your favourite hot drink. Be efficient with your writing, but be sure to include all the vital bits. You may leave out unnecessary details such as exactly how you boil the water (abstraction).

Task 3 – Do You Take Sugar?

Imagine that you are making a cup of tea or coffee for a friend. It's the first time you have made them a drink and you don't know if they take sugar. The instructions you follow would be similar to the ones in the last question, but you will need to ask them if they take sugar and then add this step if they do. You might include the instructions below:

```
……
Ask the question "Do you take sugar".
IF your friend says "Yes" THEN add sugar.
…..
```

Write some simple instructions for making a hotdog for your friend starting with a cooked sausage, a bread roll and some sauce. Include a part where you find out if your friend would like onions and add some if they do. Use the words **IF** and **THEN**.

Task 4 – Which Sauce?

Add further instructions to your hotdog recipe where you find out whether your friend wants brown sauce or red sauce. Use the words **IF**, **THEN** and **ELSE**.

Functions

A function is a section of programming code that performs a specific task. It is common to enter some data into the function (the input) and receive some different data back (the output).

Aim: *To learn about functions.*

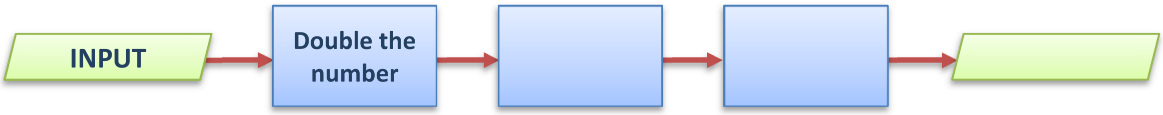
Task 1 – Mathematical Functions

a. Here is a simple function that performs a number of mathematical processes and returns an output. Complete the table showing a few selected inputs and outputs from this function.

```
Function Double_Add5_Double
    Take the input and double it
    Add 5
    Double it
End Function
```

Input	Output
1	14
2	
10	
17	
	26

b. We could show these steps in a simple flowchart. Fill in the missing steps in the flowchart below.



c. Design your own mathematical function. Draw a flowchart and work out a few input-output combinations.

d. Scientific calculators have a variety of built-in functions. Generally, you start with a number, press a function key and end up with a different number (although the actual process can vary). Use a calculator to find the output of the following functions.

Hint: *Google 'scientific calculator' if you would like to use an online calculator.*

Input	Function	Name	Output
123	x^2	Square	
1000	$\log_{(10)}$	Log (base 10)	
1000	√x	Square root	
10	!	Factorial	
35°	COS	Cosine	

Flowcharts

A flowchart can be used to represent an algorithm, showing the steps of the process as different shaped boxes connected by arrows. Flowcharts help you think through a process and make sure that all the instructions are in the correct order.

Aim: *To formally introduce the use of flowcharts.*

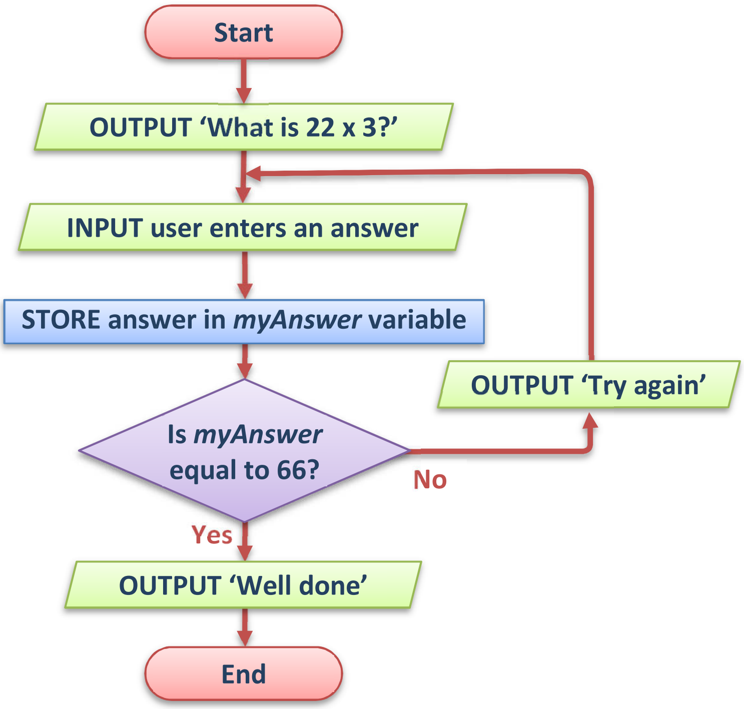
Task 1 – Flowchart Symbols

Draw connectors to match each flowchart symbol to its name and use. Look the answers up on the web if you need help, or try and guess them using the flowchart in the next task. The colours are not significant; the shapes are.

Symbol	Name	Use
⬭ •	• **Input or Output** •	• Connects the symbols and shows the direction of the flow
▱ •	• **Flow Line** •	• Shows the start and end of the process
▭ •	• **Terminal** •	• Where data is received or displayed
◇ •	• **Process** •	• A decision, usually a “Yes” or “No”
→ •	• **Decision** •	• An instruction or command

Task 2 – A Simple Flowchart

Here is a flowchart for a simple algorithm. Explain in plain English how the process works.


Start
OUTPUT ‘What is 22 x 3?’
INPUT user enters an answer
STORE answer in *myAnswer* variable
Is *myAnswer* equal to 66?
No
OUTPUT ‘Try again’
Yes
OUTPUT ‘Well done’
End

Scratch Programming

Scratch is a visual programming language used to create animations and games. It provides a great stepping stone to the more advanced world of computer programming. It can also be used for maths and science projects, animated presentations and interactive art and music.

Aim: *To learn some visual programming using Scratch.*

Task 1 – Having a Look

a. Go to https://scratch.mit.edu/ (or type 'Scratch' into Google).

b. Watch the introductory video.

c. Have a look at a few of the 'Featured Projects'.

d. Choose a simple looking project where not too much is going on and click on the 'See Inside' button. The screen should look a bit like the one on the right.

e. Click on the different *sprites* in the bottom-left section. You may notice that the scripts change. Each sprite can have different scripts (and actions) associated with it.

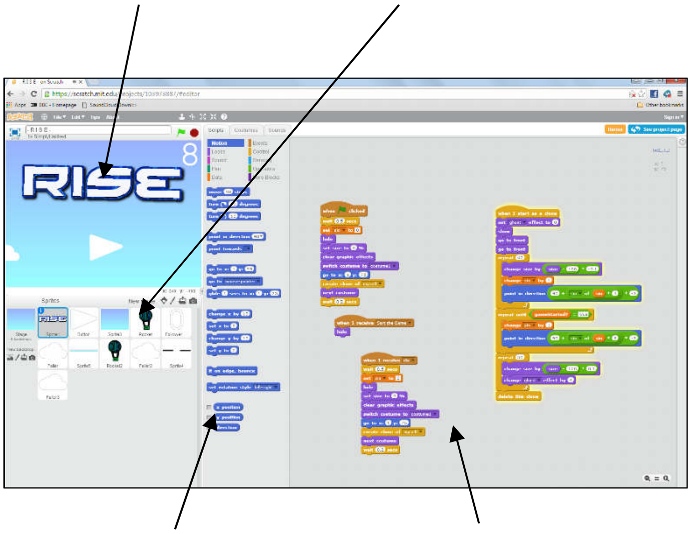
f. Have a look at the scripts and see if anything makes sense. You might recognise actions from the project such as 'move' or 'change color'.

Stage

See your project in action.

Sprites

All the things that will be visible in your project.



Script Blocks

Drag these blocks into the script area.

Script Area

Build your scripts here.

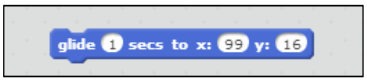
Task 2 – Having a Go

It's never too soon to jump in and have a go. You can't break anything.

a. From the homepage, click on the 'Create' link. You will start with a cat on the screen which doesn't seem to do anything.

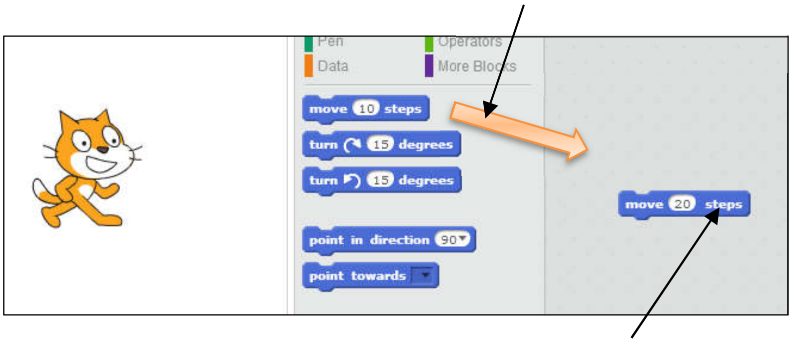
b. Drag a 'move' block into the script area as shown. Click on and change the number of steps to 20, then click on the blue area of the block. Did the cat move? Try again.

c. Try the 'glide' block. What happens when you click on the blue area?



```
glide 1 secs to x: 99 y: 16
```

1. Drag a 'move' block onto the script area



2. Click Here to see the action

Hint: *you can drag the cat to a new starting position on the stage, then try the glides again.*

Sorting

Sorting is a common task in computing. There are lots of reasons why we may want to put data in a particular order. For example, imagine looking through the contacts on your phone if they were not sorted alphabetically. It may also be necessary to keep data sorted so that the computer can use faster search algorithms, such as the *binary search*.

Aim: *To investigate how a sorting algorithm might work.*

Task 1 – The Bubble Sort

A bubble sort algorithm orders data one pair at a time, switching the data if necessary. It compares data items 1 and 2, then 2 and 3, then 3 and 4 etc., swapping the values each time if they are out of order. When it gets to the end of the data, it starts again on another 'pass'. When a pass through the data results in no change, then the data is sorted.

Complete the table below showing what happens to the 5 items of data D, C, E, A, B in a bubble sort. You should only compare the pairs of values in the green boxes, switching them if necessary while copying the data to the next column. Any data in white boxes remains the same in the next column.

Pass 1

	D	C	C	C	C
	C	D	D	D	D
	E	E	E	A	A
	A	A	A	E	B
	B	B	B	B	E
All in order?	N	N	N	N	N
Switch pair?	Y	N	Y	Y	☒
Change in pass?	Y				

Pass 2

	C				
	D				
	A				
	B				
	E				
All in order?	N				
Switch pair?	N				☒
Change in pass?					

Pass 3

All in order?					
Switch pair?					☒
Change in pass?					

Pass 4

					A
					B
					C
					D
					E
All in order?					Y
Switch pair?					☒
Change in pass?					

All in order? *Place a 'Y' in this box if all items are sorted, else enter an 'N'.*

Switch pair? *Place a 'Y' in this box if the highlighted pair need to be switched, else enter an 'N'.*

Change in pass? *Place a 'Y' in this box if there are any switches in the whole pass, else enter an 'N'.*

a. How many pair comparisons are made in each pass? ____________________

b. If 'A' started as the last item, how many passes before the data must be sorted? ____________________

c. What is the maximum number of pair comparisons required for a list of 5 items? ____________________

d. If a list had one thousand items, how many comparisons might be needed to sort the data? ____________________

A bubble sort is a relatively simple algorithm, but it can be very slow to sort large lists of data.