GraspIT ALGORITHM

PiXL Computing

Bubble Sort

A	10	20	30	40	50	60	78	80
B	15	9	22	27	14	33	30	21
C	44	30	16	51	27	28	12	6

Sort the above numbers using the **bubble sort** algorithm. Show the order of the numbers after each time you have passed through the list.

Insertion Sort

A	4	2	8	9	7	12	11	10
B	19	5	8	7	15	22	4	2
C	40	9	32	27	20	51	18	1

Sort the above numbers using the **insertion sort** algorithm. Show the order of the numbers after each time you moved/inserted a number.

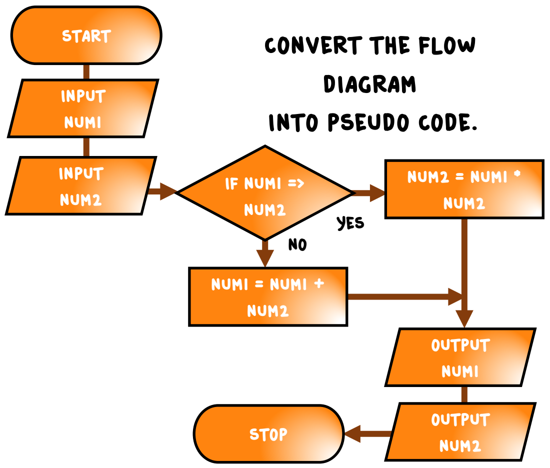
Binary Search

01101100
01101111
01100101

A	14	2	7	9	1	11	8	4	2
B	7	3	19	4	8	8	17	9	17
C	19	27	2	41	33	21	13	49	50

Search for the numbers above using the **binary search** algorithm. Show the order of the numbers after each time you have split the list into two.

Convert!



CONVERT THE FLOW DIAGRAM INTO PSEUDO CODE.

LIST THE MAIN DIFFERENCES BETWEEN THE SORT / SEARCH ALGORITHMS.

Design!

Create an algorithm for one of the situations below!
(Your algorithm can either be a **flow diagram** or **pseudo code**)
Make your algorithms as **efficient** as possible!

1) Create an algorithm which allows for any number to be entered. The number should be multiplied by 2 then output. If the number is bigger than 100 it should also print target met. If less than 100 it should print target not met.

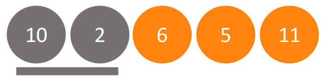
2) Create an algorithm which asks an employee how many days they work per week and how many hours they work per day. Multiply these two numbers together and then multiply it by 52 to work out how much they earn per year. Output this number.

3) Create an algorithm which will convert from miles to kilometres or vice versa. The user should be asked which calculation they want to make, the distance which they want to convert and be given the converted distance.

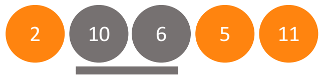
Bubble Sort - Summarise the method of a bubble sort.

A **bubble sort** works by comparing each item in the list to the item to the right of it. The item is then moved based on which is bigger / smaller.

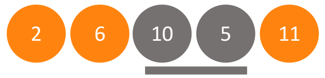
Step 1: Compare the first two numbers. Are they in order? Adjust them if not.



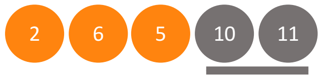
Step 2: Move to the next two numbers. Are they in order? Adjust them if not.



Step 3: Move to the next two numbers. Are they in order? Adjust them if not.



Step 4: Move to the next two numbers. Are they in order? Adjust them if not.

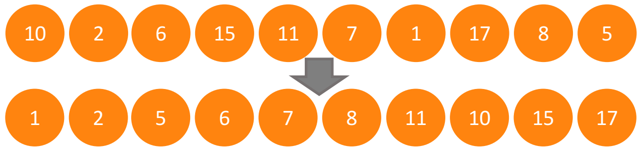


Step 5: Once you have been through the entire list, check that they are in order. If they are, great! If not, undertake the same process again until they are in order.

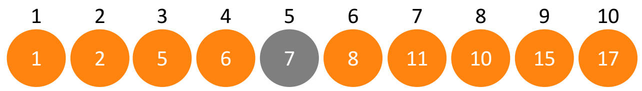
Binary Search - Summarise the method of a binary search.

A **binary search** works by repeatedly dividing the number of items by two until you are left with the item that you are searching for. We are searching for the number 2!

Step 1: Put the items into order.



Step 2: Locate the middle number (Divide the total by 2 e.g. 10/2 = 5)



Step 3: Check! Is your this number less than, equal to or greater than the number you are looking for?

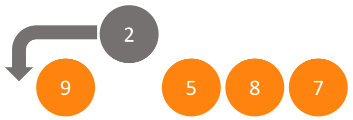
If it is greater than, you can remove all of the numbers to the right. If it is less then, you can remove all of the numbers to the left.

Repeat steps 2 and 3 until you find the number you are looking for.

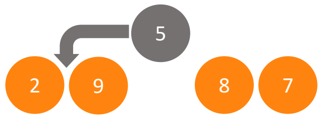
Insertion Sort - Summarise the method of an insertion sort.

The **insertion sort** works by looking at each value in turn and inserting the value into its correct place in the list.

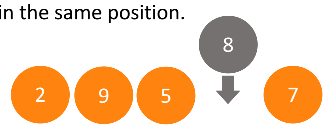
Step 1: Compare the first two items.
9 > 2 so 2 moves position.



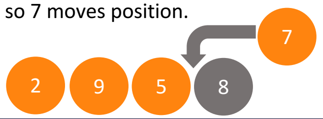
Step 2: Insert 5 into its correct position.
5 > 2 and 5 < 8 so 5 moves position.



Step 3: Insert 8 into its correct position.
8 > 5 so stays in the same position.



Step 4: Insert 7 into its correct position.
7 > 5 and 7 < 8 so 7 moves position.



PiXL Computing

GraspIT ALGORITHM

Algorithms

Explain why the order of instructions in an algorithm is important.

Explain why it is important to design/create algorithms before creating a coded solution.

Explain which search method is most efficient?

Explain which search method is least efficient?

Creating Algorithms

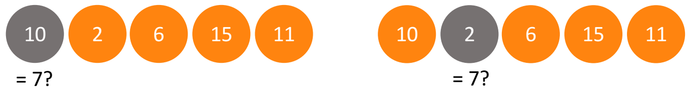
Give a **benefit** of using a flow diagram over pseudo code.

Give a **benefit** of using pseudo code over a flow diagram.

Select an algorithm from over the page. **Convert** it from a **flow diagram** to **pseudocode** or vice versa.

Linear Search - Summarise the method of a linear search.

A linear search checks item in the list to find the item that you are looking for. We are searching for the number 7!



This continues until the desired value is located!

Algorithm Efficiency

When creating algorithms it is important that they are **efficient**. This would mean using the least numbers of instructions possible. This would make the algorithm quicker to go through. This is known as **time efficiency**.