STOKE DAMEREL

Aspire Achieve Thrive

Cycle 4

Computer Science

Year 10

Name: _____

Tutor: _____

Year 10 Homework Timetable

Monday	Bedrock Learning	Ebacc Option D	Option C	Modern Britain
Tuesday	English	Tassomai	Option B	Option A
Wednesday	Hegarty	Science	Modern Britain	Option C
Thursday	Ebacc Option D	Tassomai	Bedrock Learning	Option B
Friday	Hegarty	Science	English	Option A

Tassomai - 50 questions per week
Hegarty - 4 tasks of Hegarty per week

Block A	Block B	Block C	Block D
Art	Business Studies	Art	French
Dance	Child Development	Business Studies	Geography
Drama	Catering	Geography	History
Media Studies	Computer Science	Health & Social Care	
Music	Drama	History	
Photography	Health & Social Care	Catering	
	IT	Photography	
	Media Studies	Sport	
	Sociology	Travel & Tourism	
	Sport		

Aspire | Achieve | Thrive

GREENSHAW LEARNING TRUST

Homework Plan

Week/Date	Homework Task	Examination Question Topic
Week 1 Monday 25th April	**Cornell Notes** Computational Thinking	Define the term: abstraction. Give an example of how abstraction could be used.
Week 2 Monday 2nd May	**Cornell Notes** Linear and Binary Search Algorithms	Reasons binary search cannot be used on some sets of data. Show the stages of a binary search.
Week 3 Monday 9th May	**Cornell Notes** Bubble Sort Algorithm	Show the stages of a bubble sort
Week 4 Monday 16th May	**Cornell Notes** Insertion Sort Algorithm	Show the stages of an insertion sort
Week 5 Monday 23rd May	**Cornell Notes** Merge Sort Algorithm	Show the stages of a merge sort Explain the advantage of a merge sort compared to bubble sort.
HALF-TERM		
Week 6 Monday 6th June	**Cornell Notes** Flow diagrams	Draw an algorithm as a flowchart.
Week 7 Monday 13th June	**Revision** All topics	
Week 8 Monday 20th June	**Revision** All topics	
Week 9 Monday 27th June	**Plug the gap** Areas of weakness in assessment	

Week 1: Computational Thinking (Revision Guide page 33)

Keywords	Knowledge
Algorithm - a set of instructions for completing a task. Abstraction - removing unnecessary detail from a model. Decomposition - repeatedly breaking down a problem into smaller, more solvable problems. Algorithmic Thinking - thinking about a problem logically and the steps that would be required to solve it.	The main strands of Computational Thinking are **Abstraction, Decomposition and Algorithmic Thinking**. **Abstraction** involves removing unnecessary detail from a problem. For example, the London Underground map is an abstraction of the physical tunnels, tracks and stations that make up the rail network. It only includes the detail that is required to allow commuters to travel on the network. In programming, the command **random.randint(1,6)** could be considered an abstraction of a dice roll, because it returns a number between 1 and 6. **Decomposition** involves breaking down a problem into smaller and smaller sub-programs so that each smaller program can be solved independently. In programming, one way we can decompose a problem is by using a set of sub-programs (in Python, we use **def** to define a subprogram) to 'break down' different parts of a program. An **algorithm** is a set of instructions for completing a task. For example: A recipe for baking a cake, instructions for building a lego model. **Algorithmic Thinking** involves thinking about a problem logically and defining the steps that would be required to solve them.

Week 2: Linear Search and Binary Search Algorithms (Revision Guide page 36)

Keywords

Linear Search - checks each item in turn until the item is found, or all the items have been checked.

Binary Search - A search strategy that uses a 'divide and conquer' approach by continually reducing the size of the list of items, until the correct item is found, or all items have been checked.

Ordered - items are arranged in an order. For example: Ascending from A-Z or Descending from 9-0.

Knowledge

Binary Search Algorithm:	**Linear Search Algorithm:**
1. Find the middle item in a ordered list 2. If this is the item you're looking for, stop the search. 3. If not, compare the item you're looking for to the middle item in the list. If it comes before the middle item, remove the second half of the list. If it comes after the middle item, remove the first half of the list. 4. You'll be left with a list that is half the original size. Repeat steps 1 to 3 until the item is found or until the entire list has been checked.	1. Look at the first item in the unordered list 2. If this is the item you're looking for, then stop the search. 3. If not, then look at the next item in the list. 4. Repeat steps 2 and 3 until the item is found, or each item in the list has been checked.
A binary search is more efficient (faster to complete) than a linear search,.	Linear search is much simpler to program than a binary search.
A binary search can only be performed on an ordered list. If the list is not sorted, it will need to be sorted first.	Linear search can be performed on ordered and unordered lists.
In general, takes less 'iterations' to complete a binary search so is faster to complete.	Can be used on very small lists (less than about 5-10 items)

Week 3: Bubble Sort Algorithms (Revision Guide page 46)

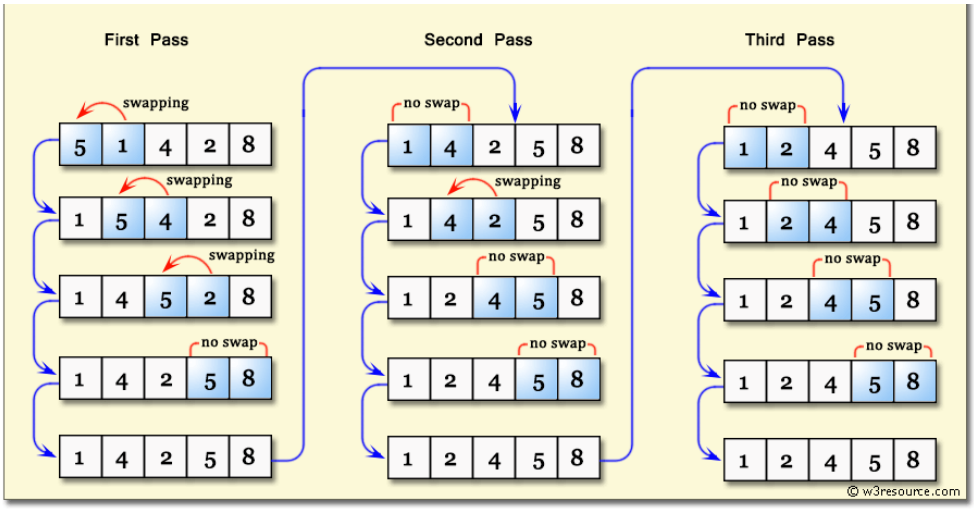
Knowledge

Bubble Sort is a type of sorting algorithm, used to sort an unordered list of items.

Bubble Sort Algorithm

1. Look at the first two items in the list.
2. If they're in the right order, don't do anything.
3. Move on to the next pair of items (the 2nd and 3rd entries) and repeat step 2
4. Repeat step 3 until you get to the end of the list - this is called one pass. The last item will now be in the correct place, so is not included in the next pass.
5. Repeat steps 1 to 4 until there are no swaps in a pass.

Worked Example



In your Revision Notes: Practice using the bubble sort algorithm for different groups of values. You can also sort text. Often this is done using the ASCII decimal value for each character, therefore A is less than B.

Week 4: Insertion Sort Algorithm (Revision Guide page 48)

Knowledge

Insertion sort is a sorting algorithm that takes each item in turn and puts it in the right position, starting with the first item in the list.

Insertion Sort Algorithm

1. Look at the second item in a list.
2. Compare it to all items before it (in this case, just the first item) and insert it into the right place.
3. Repeat step 2 for the third, forth, fifth, etc. items until the last number in the list has been inserted into the correct place.

Worked Example

In your Revision Notes:

Practice using the insertion sort algorithm for different groups of values. You can also sort text. Often this is done using the ASCII decimal value for each character, therefore A is less than B. Check out https://www.asciitable.com/ for more information.

Week 5: Merge Sort Algorithm (Revision Guide page 47)

Knowledge

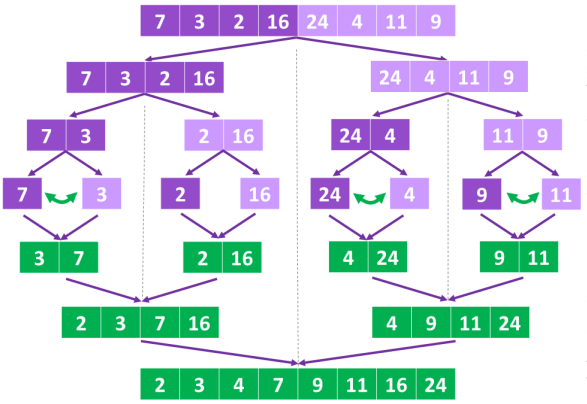
Merge sort is an example of a divide and conquer algorithm and takes advantage of the following two facts:

- Small lists are easier to sort than large lists
- It's easier to merge two ordered lists than two unordered lists.

Merge Sort Algorithm

1. Split the list into half (the smaller lists are called sub-lists) - the second sublist should start at the middle item.
2. Keep repeating step 1 on each sub-list until all the lists contain only one item.
3. Merge the pairs of sub-lists so that each sub-list has twice as many items. Each time you merge sub-lists, sort the items into the right order.
4. Repeat step 3 until you have merged all the sub-lists together.

Worked Example



https://www.asciitable.com/)

In your Revision Notes:

Practice using the merge sort algorithm for different groups of values. You can also sort text. Often this is done using the ASCII decimal value for each character, therefore A is less than B.

Week 6: Flow Diagrams (Revision Guide page 44)

Knowledge

Algorithms can be noted in a range of ways. One way to represent a set of instructions is through using a flow diagram. This uses arrows to show the flow of a program from start to end. Standard symbols are used in flowcharts.

Start / Stop	The beginning and the end of the algorithm are put into boxes with rounded corners. It's good practice to include the name of the program / subroutine. For example: Start DiceGame, End DiceGame	Decision	Decisions should result in a True / False result. If you intend to use an if statement when coding the algorithm, this should indicate that you need a decision in your flow diagram.
Input/Output	Anything that's put into or taken out of the algorithm goes in a parallelogram box. For example, if you need an input() or print() statement when you code your algorithm.	Sub Routine	If you want to 'call up' a sub-routine or sub-program as part of your main program.
Process	General instructions, processes and calculations go in rectangular boxes. For example, assigning a variable (a = 5) or multiplying two numbers (a = b * c)	→	Arrows connect boxes and should always show the flow of a program

Week 7 and 8: Preparing for Assessment

Self-quiz the knowledge covered in Weeks 1 - 6

WEEK 1

Date……

Elliott plays football for OCR FC. He wants to create a program to store the results of each football match they play and the names of the goal scorers. Elliott wants individual players from the team to be able to submit this information.

i. **Define** what is meant by abstraction.

[2 marks]

ii. **Give one example** of how abstraction could be used when developing this program.

[1 mark]

WEEK 2

Date……

OCR Land is a theme park aimed at children and adults. OCR Land regularly emails discount codes to customers.

A list of valid discount codes is shown below.

[NIC12B, LOR11S, STU12M, VIC08E, KEI99M, WES56O, DAN34S]

i. **State one reason** why a binary search could not be used to find a discount code within this data.

[1 mark]

ii. **Give** the name of one searching algorithm that would be able to be used with this data.

[1 mark]

Willow has created a hangman program that uses a file to store the words the program can select from. A sample of this data is shown below:

amber	house	kick	moose	orange	range	tent	wind	zebra

Show the stages of a binary search to find the word 'zebra' when applied to the data shown.

[4 marks]

WEEK 3

Date……

Willow has created a hangman program that uses a file to store the words the program can select from.
A sample of this data is shown below:

crime	bait	fright	victory	nymph	loose

Show the stages of a bubble sort when applied to the data shown

[4 marks]

WEEK 4

Date…….

The following names of students are stored in an array with the identifier studentnames.

studentnames = ["Rob", "Anna", "Huw", "Emma", "Patrice", "Iqbal"]

The names of students are sorted into ascending alphabetical order using an insertion sort.

Complete the following diagram to show the stages an insertion sort would take to complete this task.

Each row represents one pass of the insertion sort algorithm. You may not need to use all empty rows.

Rob	Anna	Huw	Emma	Patrice	Iqbal

[4 marks]

WEEK 5

Date…….

A library gives each book a code made from the first three letters of the book title in upper case, followed by the last two digits of the year the book was published.

For example, “Poetry from the War”, published in 2012 would be given the code POE12.
The library sorts their books based on the book code.

i. **Show the steps** that a merge sort would take to put the following list of book codes into ascending alphabetical order (from A to Z).

POE12 , BAC97 , FLY77 , JAV16 , TAL86 , AND18 , ZAR09 , HOP86

[4 marks]

ii. **Explain** one advantage of a merge sort compared to a bubble sort.

[2 marks]

WEEK 6

Date…….

A vending machine has the following options available.

Item Code	Item name	Price
A1	Crisps, bacon flavour	£0.75
A2	Crisps, salted	£0.75
B1	Chocolate bar	£0.90
C1	Apple pieces	£0.50
C2	Raisins	£0.85

Users insert coins into the vending machine and then enter the two character item code of their selection. If the user has inserted enough money, the vending machine will release the chosen item and output any change required. If the user enters an invalid item code then a suitable error message is displayed.

Draw the vending machine algorithm in the part above as a flowchart. (Use additional paper if required)

[5 marks]

STEP 2: CREATE CUES

What: Reduce your notes to just the essentials.

What: Immediately after class, discussion, or reading session.

How:

- **Jot down key ideas, important words and phrases**
- **Create questions that might appear on an exam**
- **Reducing your notes to the most important ideas and concepts improves recall. Creating questions that may appear on an exam gets you thinking about how the information might be applied and improves your performance on the exam.**

Why: Spend at least ten minutes every week reviewing all of your previous notes. Reflect on the material and ask yourself questions based on what you've recorded in the Cue area. Cover the note-taking area with a piece of paper. Can you answer them?

STEP 1: RECORD YOUR NOTES

What: Record all keywords, ideas, important dates, people, places, diagrams and formulas from the lesson. Create a new page for each topic discussed.

When: During class lecture, discussion, or reading session.

How:

- **Use bullet points, abbreviated phrases, and pictures**
- **Avoid full sentences and paragraphs**
- **Leave space between points to add more information later**

Why: Important ideas must be recorded in a way that is meaningful to you.

STEP 3: SUMMARISE & REVIEW

What: Summarise the main ideas from the lesson.
What: At the end of the class lecture, discussion, or reading session.
How: In complete sentences, write down the conclusions that can be made from the information in your notes.
Why: Summarising the information after it's learned improves long-term retention.

Date / / **Topic**

WEEK 1

Questions	Notes

Summary

Date / / **Topic**

WEEK 2

Questions	Notes

Summary

Date / / **Topic**

WEEK 3

Questions	Notes

Summary

Date / / **Topic**

WEEK 4

Questions	Notes

Summary

Date / / **Topic**

WEEK 5

Questions	Notes

Summary

Date / / **Topic**

WEEK 6

Questions	Notes

Summary

Revision Page

Revision Page

Revision Page

Revision Page

Revision Page

Revision Page

Revision Page

Revision Page
