GCST

1.4.2 DATA STRUCTURES

TOPIC WISE EXAM QUESTIONS

ANSWERS

A-LEVEL OCR

GCSECOMPUTERSCIENCETUTOR.COM

2	(a)		• value == "E"
			• value == "S"
			• (numone+numtwo)
			• value
			• "E"
				5	All string values must be in quotes. Allow single or double quotes.
					
					Don't allow single = for MP1&2. Penalise once and FT for multiple occurrences.
					
					Case needs to match that used in the question
					
					Needs to have brackets for MP3
					
					do
					  value = input("Enter a value")
					  if value == "E" then
					    num = numbers.pop()
					    print(num)
					  elseif value == "A" or value == "S" then
					    numone = numbers.pop()
					    numtwo = numbers.pop()
					    if value == "A" then
					      numbers.push(numone + numtwo)
					    elseif value == "S" then
					      numbers.push(numtwo - numone)
					    endif
					  else
					    numbers.push(value)
					  endif
					until value == "E"

2	(b)	(i)	• 8, 7 • 15 • 15,6	3	One mark per stack diagram
2	(b)	(ii)	• 12 • 7 • 15	3	
2	(b)	(iii)	• S causes the two values inputted to be popped and only one value to be pushed back // 4 and 2 are popped and 2 is pushed • A causes an attempt to pop **two values but only one present / not two values there** • Causing a **stack** underflow	3	
2	(c)	(i)	• Stack is LIFO / FILO • Queue is FIFO / LILO • Stack uses one pointer (for head) • Queue uses two pointers (head and tail) • Stack, data is popped/pushed from the top • Queue, data is dequeued from the from the start and enqueued onto the back // a queue can be circular	2	Mark in pairs Accept descriptions of LIFO / FIFO for MP1 and 2
2	(c)	(ii)	• Array is of fixed/defined size // static • List size can be changed // no defined size // dynamic • Array holds data of single data type • List can hold data of multiple / different types	2	Mark in pairs
2	(c)	(iii)	• A tuple cannot be changed **at runtime** // a tuple is immutable	1	
2	(c)	(iv)	• **Go to** the first position indicated by the start pointer • From the first position, **read** the next pointer value… • …follow this pointer value and **access** the data item	3	Accept answers relating to locations given by pointers Allow acceptable diagram illustrating the same points

2	(a)	(i)	☑ First In First Out	AO1.1 (1)	
2	(a)	(ii)	Ben | Sundip | Tom | Charlie | Ling | Sara | | 1 Mark: Adding, Charlie, Ling and Sara in correct order 1 Mark: Exclusively removing Alex and Kofi.	AO2.1 (2)	Accept: | | Ben | Sundip | Tom | Charlie | Ling | Sara

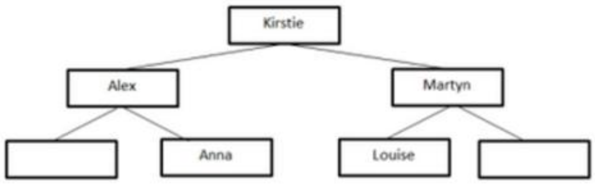
2	(b)	(i)	`function pop()`
			`   if top == 0 then`
			`      return -1`
			`   else`
			`      item = items[top]`
			`      top = top - 1`
			`      return item`
			`   endif`
			`end function`
				AO3.2
				(4)	

2	(b)	(ii)	• Correctly declaring the function **reverse** to include passing in **name** as a parameter
			• Correct logic to calculate the number of pushes required
			• Correct use of a loop to push all characters onto the stack separately
			• Creating a local variable **reverseName** to hold the reversed string
			• Correct use of a loop to pop all characters from the stack (in the reverse order)
			• Correct logic to add each popped character to the **reverseName** variable
			• Correctly returning the **reverseName** variable
				AO3.1
				(3)
				AO3.2
				(4)
					Example solution
					`function reverse(name)`
					`   reverseName = ""`
					
					`   for nameCount = 0 to name.Length-1`
					`      theStack.push(name[nameCount])`
					`   next nameCount`
					
					`   for nameCount = 0 to name.Length-1`
					`      reverseName = reverseName + theStack.pop()`
					`   next nameCount`
					
					`   return reverseName`
					`end function`
					
					Give full marks for alternative solutions that would work fully.
					
					Allow FT for any duplicate identifiers named incorrectly or using the incorrect case

2021

7	(a)	(i)	Stack	1 AO1.1	Correct answer only

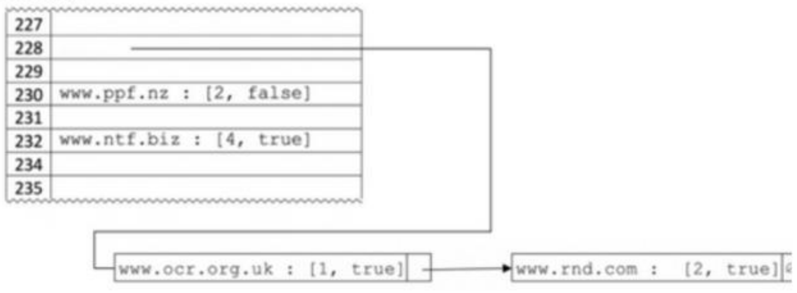
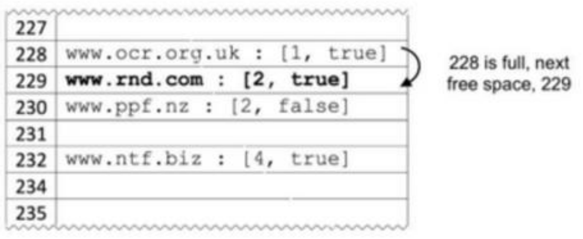
		(ii)	• Input name from user
			• Check if stack is full e.g.(top <=5)
			• If not, update top pointer
			• Correctly push name to index referenced by top
				4
				AO3.2
					<u>Example answer</u>
					`new = input("enter a name : ")`
					`if top <=5 then`
					`    top = top + 1`
					`    wNames[top] = new`
					`end if`

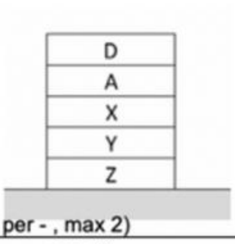
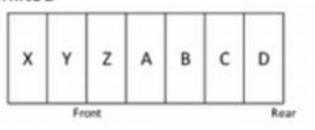
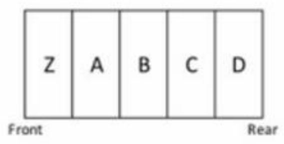
	(b)		1 mark per name inserted in correct place in diagram	4 AO2.1	
		(ii)	• Compare with Kirstie – Zoe is larger so go right • Compare with Martyn - Zoe is larger so go right • No right element so stop/not found	3 AO2.2	Allow FT from b(i)
		(iii)	• Binary tree more efficient than linked list • Do not need to check every value / tree removes half values each time	2 AO1.2	Allow reference to big O for second mark. Linked List O(n), Binary Tree O(log n)
		(iv)	• Binary tree less efficient than hash table • Hash table can find data immediately / without checking other values.	2 AO1.2	Allow reference to big O for second mark. Binary tree O(log n) Hash table O(1)

2019

5	a		tuple / record / list	1 AO2.1	Don't accept array
	b		228	1 AO2.1	cao

	c		– Removes/ignores characters up to and including first dot
			– Removes/ignores characters including and after second dot
			– Converts characters in variable to uppercase
			– Totals the ASCII values of the relevant characters.
			
			Up to 1 mark for…
			– Sensible variable names.
			– Sensible indentation
			– Useful comments.
			
			NB Don't penalise twice. If candidate hasn't removed/discounted the right characters they may lose mark points 1 and/or 2. They can still access mark points 3 and 4
				5
				AO3.2
					`function hash(siteName)`
					`  //remove up to and including first dot.`
					`  firstDot=locate(siteName,".")`
					`  siteName=siteName.substring(firstDot+1,siteName.length-firstDot -1)`
					
					`  //remove second dot and after`
					`  secondDot=locate(siteName,".")`
					`  siteName=siteName.substring(0,secondDot)`
					`  siteName=upper(siteName)`
					`  value=0`
					`  for i=0 to siteName.length-1`
					`  value=value+asc(siteName.substring(i,1))`
					`  next i`
					` return value`
					`endfunction`

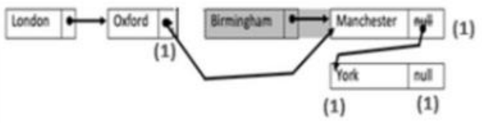
d		– rnd.com would cause a collision with ocr.org.uk/would has to the same position as ocr.org.uk (228) – Linear probing could be used – Move through the structure one space at a time – …to find the next free space/229 – Chaining could be used – Each location points (to the start of) a <u>linked</u> list. – The new item is added to the end of the linked list/free. – points to an overflow area – The new item is stored with the other values in the same area Accept a separate or annotated diagram showing a method on given example (1 per -, max 4)	4 AO1.2	Example diagrams Chaining  227 228 229 230 www.ppf.nz : [2, false] 231 232 www.ntf.biz : [4, true] 234 235 www.ocr.org.uk : [1, true] www.rnd.com : [2, true] Linear Probing  227 228 www.ocr.org.uk : [1, true] 229 www.rnd.com : [2, true] 230 www.ppf.nz : [2, false] 231 232 www.ntf.biz : [4, true] 234 235 228 is full, next free space, 229
e		**Mark Band 3–High Level (9-12 marks)** The candidate demonstrates a thorough knowledge and understanding of storing and retrieving data from hash tables and linked lists. The material is generally accurate and detailed. The candidate is able to apply their knowledge and understanding directly and consistently to the context provided. Evidence/examples will be explicitly relevant to the explanation. The candidate is able to weigh up both sides of the argument which results in a supported and realistic judgment as to which data structure is suitable. There is a well-developed line of reasoning which is clear and logically structured. The information presented is relevant and substantiated. **Mark Band 2-Mid Level (5-8 marks)** The candidate demonstrates reasonable knowledge and understanding of storing and retrieving data from hash tables or linked lists; the material is generally accurate but at times underdeveloped. The candidate is able to apply their knowledge and understanding directly to the context provided although one or two opportunities are missed. Evidence/examples are for the most part implicitly relevant to the explanation. The candidate makes a reasonable attempt to come to a conclusion showing some recognition of influencing factors that would determine which data structure is suitable. There is a line of reasoning presented with some structure. The information presented is in the most part relevant and supported by some evidence. **Mark Band 1-Low Level (1-4 marks)** The candidate demonstrates a basic knowledge of storing and retrieving data from hash tables or linked lists; the material is basic and contains some inaccuracies. The	12 AO1.1 (2) AO1.2 (2) AO2.1 (3) AO3.3 (5)	Searching of a Linked list involves starting at the first node and following the pointers until either the desired value is found, or the end of the list is reached, meaning the item isn't in the list. The bigger the linked list grows, the longer it takes to search. If a linked list doubles in size it will, on average, take twice as long to search. A list of size n takes on average n/2 checks. In Big O this is O(n), or linear complexity. Searching of a hash table requires the key to be hashed and the correct location accessed. The time this takes is largely dependent on the time to create the hash. If we ignore collisions, the time to find an item will stay the same regardless of the size of the white list. In other words it has O(1) or constant complexity. Unfortunately as the white list grows collisions become more likely. Linear probing and chaining means that once a location has been found the time taken grows linearly with the number of collisions that have occurred for that location, Nonetheless this is still going to perform significantly better than a linked list. If items are added to the end of the linked list then if the location of the last node is stored, that location can be ready made to point at the new item. The time to add items is constant. If they are added in some sort of order then the time to add items grows linearly due to the time spent searching for the right position. (Storing in order has the advantage that it is if an item isn't in the list this can be deduced once its location is passed, rather than waiting until the end.) Adding items to a hash table involves hashing the key and placing it in the correct location. This takes a constant amount of time.. ..unless there are collisions then there is an overhead which grows with the number of collisions for that location. Overall a hash table is likely to be the best option (assuming it has enough space and a good hashing algorithm which produces a hash quickly and with few collisions). It will give very consistent performance even as the whitelist grows. *NB Big O Notation isn't necessary for full marks as it is paper 2 content (though many of the more able candidates are likely to include it). The question is assessing candidate's knowledge of traversing and adding to the two data structures and their ability to analyse this to determine their suitability for the scenario.*

4	a		A queue	1 (AO1.1)	
	b		- D at top of stack with A directly below it	2 (AO2.2)	Allow new drawing or amendment of original.
			- X,Y,Z directly below A (with no other entries)		
					
			(1 per - , max 2)		
	c		- `pop()`	3 (AO3.1)	
			- `pop()`		
			- `push("A")`		
			(1 per - , max 3)		
	d		- Z the front element AND correct front pointer	2 (AO2.2)	Allow X and Y to still be visible if front pointer has been shifted
			- Followed directly by ABCD AND correct rear pointer		
			(1 per - , max 2)		
					

6	a		- A data structure/holds multiple pieces of data…	2 (AO1.2)	
			- Has a single identifier		
			- Elements are accessed by an index		
			- Holds data of the same data type		
			- Elements are stored contiguously in computer memory		
			(1 per - , max 2)		
	b		- Declaration of list/array.	5 (AO3.2)	<u>Array Version</u>
			- for loop which runs ten times.		`array names[9]`
			- inputting name to correct location each iteration.		`for i = 0 to 9`
			- for loop/while loop which outputs each name.		`    names[i]=input("Enter a name: ")`
			- names are formatted with numbers 1-10 and a dot preceding each one.		`next i`
			(1 per - , max 5)		`for i= 0 to 9`
					`    print((i+1)+". "+names[i])`
					`next i`
					List Version
					`names = []`
					`for i = 0 to 9`
					`    names.append(input("Enter a name: "))`
					`next i`
					`for i= 0 to 9`
					`    print((i+1)+". "+names[i])`
					`next i`
					Allow 0 or 1 based array, provided code is consistent.

GCST

2017

2	a	i	A dynamic/data structure (1) Each node/item consists of data and pointer (1) Pointer gives location of next node. (1)	3 (AO1.2)	Accept 'element' instead of 'node/item'
		ii	Description can be written: - Oxford pointer changed to bypass Birmingham and point to Manchester. (1) - A node is created holding the data York/York is placed is next free space/node/item (1) - Manchester remains in original position and pointer changed to point to the York node. (1) - The York node points to null (or terminator). (1) OR via diagram eg.:  London Oxford Birmingham Manchester null York null (1) (1) (1) (1)	4 (AO2.1)	On diagram don't penalise if the pointer from Birmingham is left intact. It should be clear in both diagram and text that Oxford no longer points to Birmingham. In diagram solution, London, Oxford and Manchester must remain in the same positions.
	b		A linked list requires every node to be checked (until the desired record is found). (1) A linked list will take longer to search (as more nodes are added). (1) A hash table enables direct access to the location of the record. (1) A hash table will take the same time to search (as more nodes are added)/It takes no longer as more records are added. (1)	4 (AO1.2 - 2 marks AO2.2 - 2 marks)	Some candidates may talk about time complexity: linked lists being linear/O(n) and hash table being constant/O(1) Accept these as points 1& 2 and 3 & 4 conjoined i.e. full marks.

2016

AS - Level

7			• Prints receipt with item name and price on each line. (AO3.2) • Applies a 10% discount to gardening purchases. (AO3.2) • If decorating spend is £20 or more. (AO3.2) • Displays each discount on the receipt. (AO3.2) • Displays the correct total. (AO3.2) • Correct addressing of a 2D array (A02.1)	6 **A02.1** **(1)** **AO3.2** **(5)**	Example

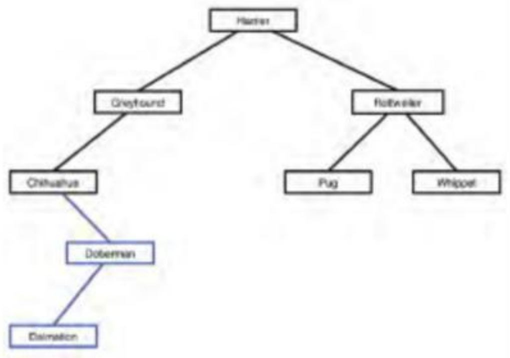
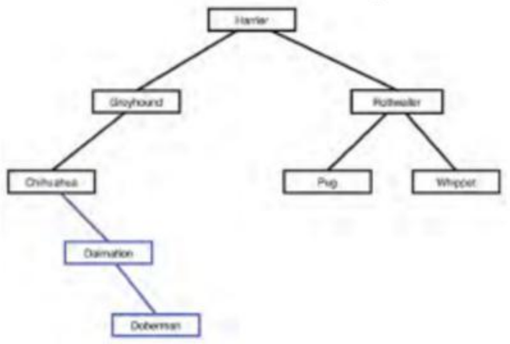
```
decoratingSpend = 0.0
for i = 0 to purchases[].size()
   if purchases[i,1] == "Decorating" then
        decoratingSpend = decoratingSpend +
                     float(purchases[i,2])
   endif
next i

total = 0.0
disc = 0.0

for i = 0 to purchases[].size()
   print(purchases[i,0] + " £" + purchases[i,2])
   total = total + float(purchases[i,2])
   if decoratingSpend >= 20 AND purchases[i,1] ==
                                "Gardening" then
      disc = round(float(purchases[i,2]) * 0.1, 2)
      print("-£" + str(disc) + " discount")
      total = total - disc
    endif
next i
print("------------")
print("TOTAL: £" + str(total))
```

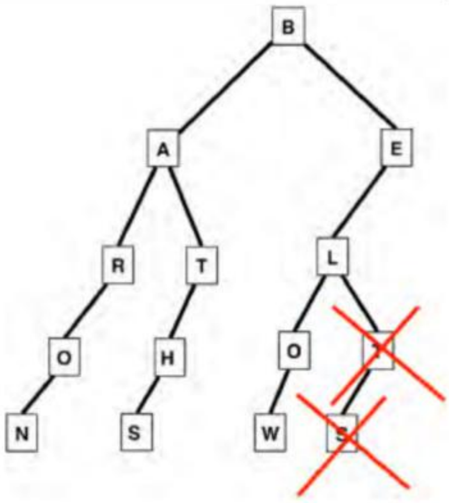
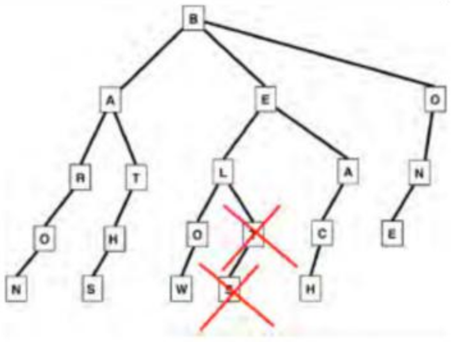
1			• Accounts.doc, budget.xls (1). • Followed by beach.jpg, sunset.jpg, hotel.jpg (in any order) (1). • Followed by tournament.xls (1).	3	For 3 marks. If answer includes directory names ignore the directories and just mark order of files.
			Total	**3**	
2	a		• Graph (1).	1	For 1 mark. Accept 2D array.

	b	i	• Creates a variable to represent total cost and initialises it to 0 (1).
			• Iterates up to the penultimate item of array (1).
			• Adds to the total cost … (1).
			• … Uses the correct arguments in the tripCost function (1).
			• Outputs the total cost formatted with a £ prefix (1).
				5	For 5 marks – 1 mark for each correct step in process.
					
					Any program that has the functionality specified in the question should receive full marks.
					
					Example:
					
					`totalCost=0`
					`for i=0 to cities.Length-2`
					`totalCost=totalCost+`
					`tripCost(cities[i],cities[i+1])`
					`next i`
					`print("£"+totalCost)`

		ii	• A linked list is a dynamic data structure (1) whereas an array is static (1). • An array can have any element accessed directly (i.e. random access) (1) whereas a linked list needs to be traversed until the desired element is found (1). • Contents of an array are stored contiguously in memory (1) whereas the contents of a linked list may not be (1).	2	Up to 2 marks for a valid description.

	c		• Takes in code of airport (1).
			• Iterates through the array (1).
			• Checks the value of the code column at each iteration (1).
			• To see if it is equal to code given (1).
			• When it is, it takes the airport name from the name column (1).
			• And prints it to the screen (1).
				6	For 6 marks – 1 mark for each correct step in process.
					
					Any program that has the functionality specified in the question should receive full marks.
					
					Array could be 0 or 1 based.
					
					Examples include:
					
					`code=input("Please enter code")`
					`i=0`
					`while airports[1,i]!=code`
					`    i=i+1`
					`endwhile`
					`print("The airport is: "+airports[2,i])`
					
					OR
					
					`code = input("Please enter code")`
					`name=""`
					`for i=0 to 7`
					`    if airports[1,i]==code then`
					`        name=airports[2,i]`
					`    endif`
					`next i`
					`print("The airport is: "+name)`

3	a		 Harrier Greyhound Rottweiler Chihuahua Pug Whippet Doberman Dalmation Doberman in correct position (1) Dalmatian in correct position (1) (Allow FT if first mark is incorrect)	2	Allow one mark if added in wrong order.  Harrier Greyhound Rottweiler Chihuahua Pug Whippet Dalmation Doberman
	b		Pug > Harrier (go right) (1) Pug < Rottweiler (go left) (1) Found Pug (1)	3	
	c		Spaniel > Harriet (go right) (1) Spaniel> Rottweiler (go right) (1)Spaniel < Whippet, no child node so Spaniel is not in tree (1)	3	

6		i	- T and S removed /T removed/Link between L and T removed… - …No further nodes removed (1 Mark per -, Max 2)	2 (AO2.1)	 B A E R T L O H O T N S W S
		ii	- BEACH added - BONE added (1 Mark per -, Max 2)	2 (AO2.1)	 B A E O R T L A N O H O T C E N S W S H Whether branches point left or right or order of branches is irrelevant. As long as branches form the words without unnecessary repetition of nodes, award the marks. **Examiner's Comments** Invariably, all candidates fared well on both parts of this question.

If you found this useful, drop a follow to help me out!

THANK YOU!

GCST

GCSECOMPUTERSCIENCETUTOR.COM