



PRIFYSGOL CYMRU

Y Drindod Dewi Sant

UNIVERSITY OF WALES

Trinity Saint David

Examination Answer Book

24 Pages

Instructions to candidates

1. Please give the information requested in the boxes below.
2. Enter your name in the space provided to the right and seal to hide your name.
3. The answers must be **legibly** written. The Examiners can either ignore illegible script or have it typed and the expense of typing charged to the candidate.
4. Write your answers in ink.
5. Begin each answer on a fresh page. Do all rough work in the answer book and cross it through.
6. No part of this book is to be torn out; it must be handed in complete.
7. If more books than one are used, fasten them together using the tags available from the invigilator(s).
8. **Every book must be handed in. In no circumstances may answer books, used or unused, be taken out of the examination venue.**

1	Student Number	P148207
2	Title of Examination Paper	Algorithms & Data Structures
3	Module Code	M5X01023
4	Date and Time of Examination	11:00 01/02/2018
5	Examination Venue	MH312
6	Seat Number	

7	Number of Answer Books handed in	1
---	----------------------------------	---

List the numbers of the questions you have attempted	Office Use
1	22/25
2a	} 18/20
2b	
2c	
2d	} 20 1/2
3a	
3b	
3c	25
	8 1/2
Total	

Please complete this section and SEAL before the examination starts.

(Block Capitals) Forename(s)

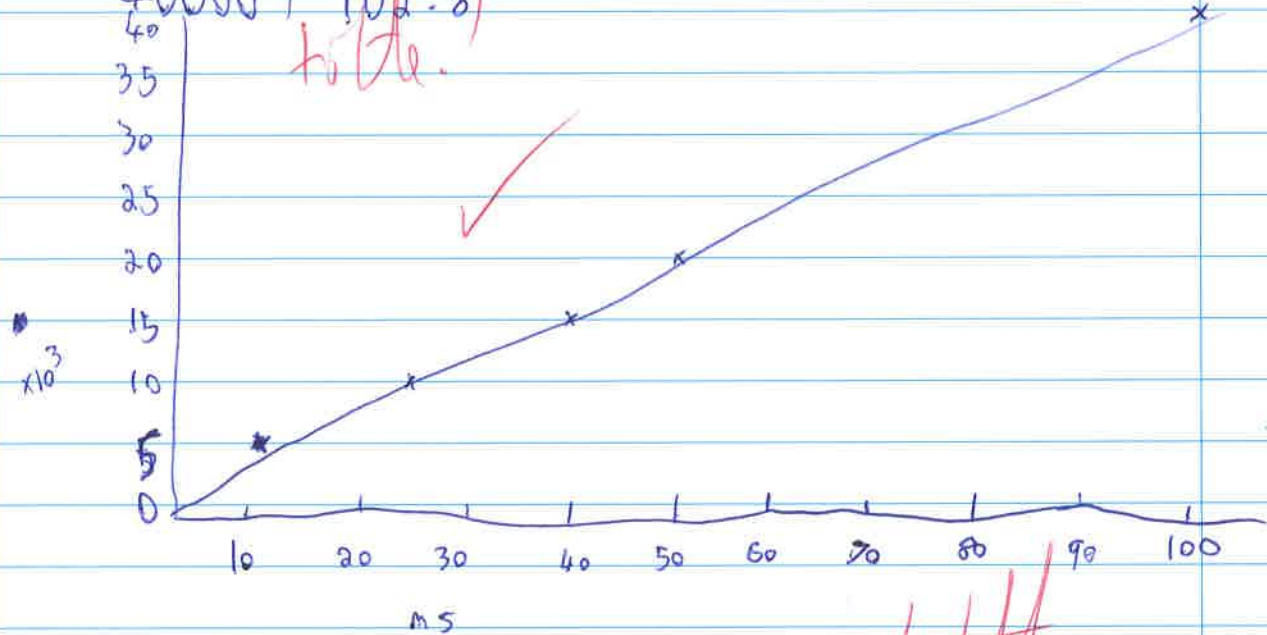
Surname

(Gweler yr ochr arall am y fersiwn Cymraeg)

Func. A

N	Time (ms)
5000	12.8
10000	25.8
15000	39.4
20000	51.1
40000	102.8

little.



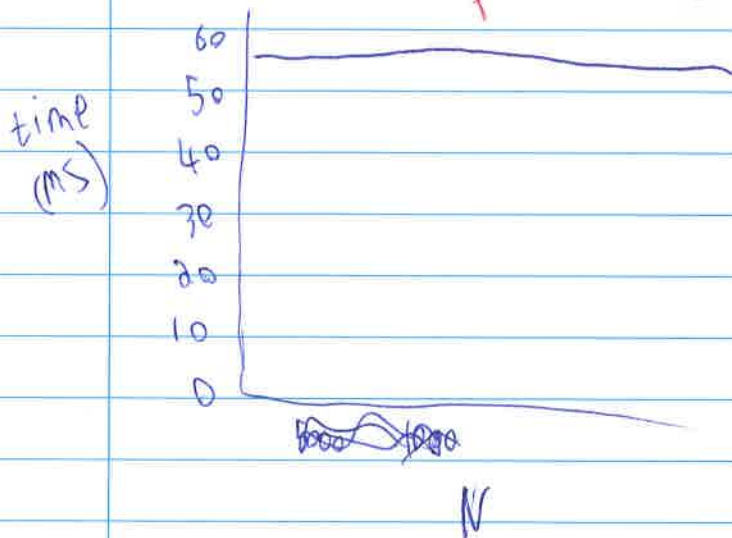
a little
memory

$O(n)$

FUNC 2

N	time (ms)
5000	57.5
10000	57.6
15000	57.3
20000	56.9
40000	57.1

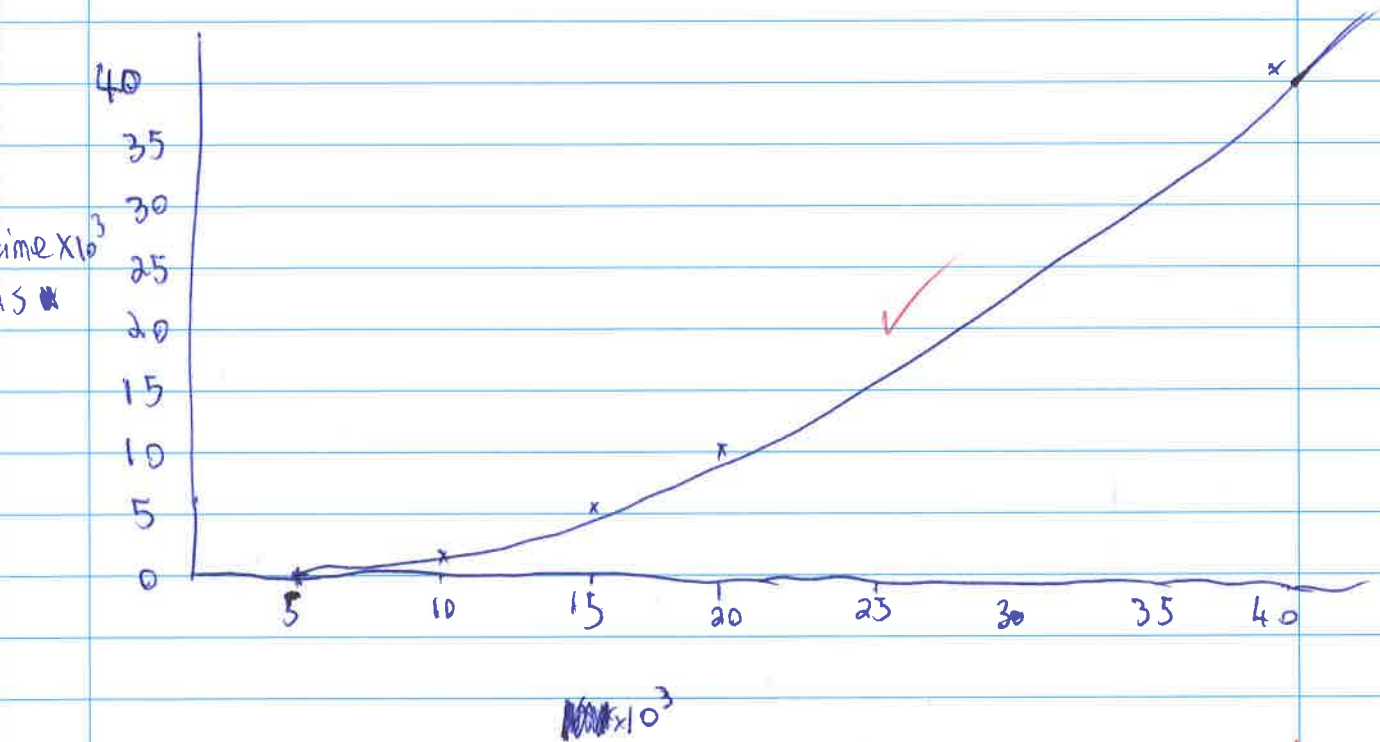
little! print where



Func 3

N	time (ms)
5000	675
10000	2669
15000	6008
20000	10664
40000	42601

quadratic



$$O(n^2)$$

see graph
page

22/25

22/25

an dth relationship
to compilation
must be

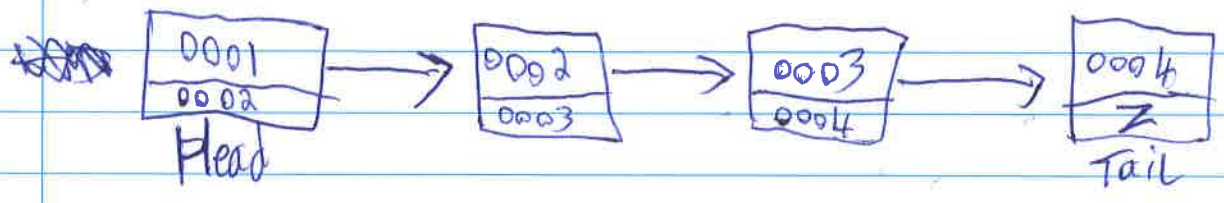
2a- static memory allocation is where an amount of memory is reserved for a process, but this amount ~~is~~ never changes regardless of usage or need.

Dynamic memory allocation is much the same except the amount of memory reserved can be changed by the program in order to correctly meet the needs of the process.

hint.

2/4

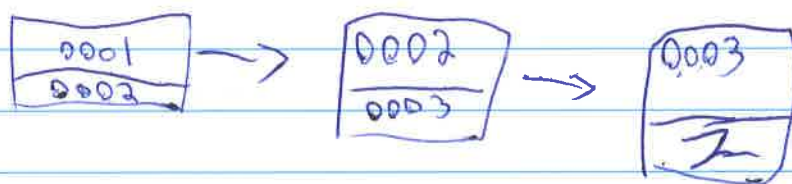
2b- A linearly Linked List is a data structure used to maintain a collection of objects in ~~an~~ an organised manner. A linked list consists of ~~one~~ head and tail nodes, these delimitate the start and end points of the linked list, and any number of data nodes between them. Each node points to the ~~the~~ address of the next node until the tail is reached as this will point to a known terminating value.



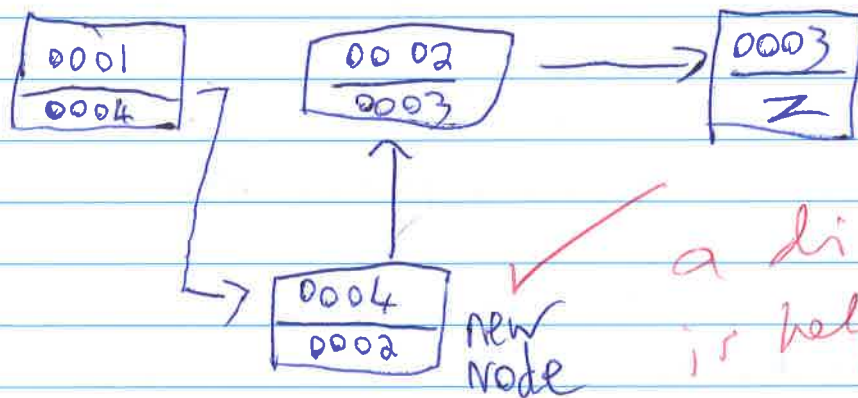
The linked list has a number of essential operations, such as adding a node, removing a node and traversal.

Traversal is simply observing the current node and evaluating the next nodes reference value from the current node then setting the current as previous and the ~~new~~ node at address to current. This allows the list to be traversed and data retrieved.

Adding a node at a point means creating an empty node, setting the current nodes next node address as the new nodes ~~node address~~ next node address pointing the current nodes next node address to the address of the new node.

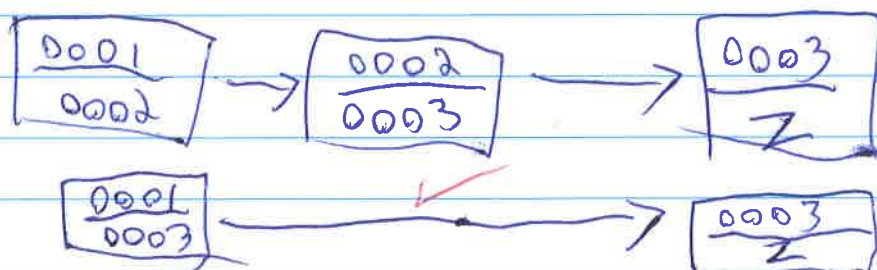


but argue for your choice



a diagram is helpful!

removing a node is much the opposite, the node before the removed one has its next node address set to that of the current node to be deleted.



2c. First ^{class} ~~an object~~ for the list item would be needed ~~the~~ so list items can be created

List Item

```
{  
  char* Address
```

• some class My Object

```
  char* Next Address  
}
```

← This is where the data is stored

When instantiating the list a head and tail would need to be created. The tail would have some sort of terminating value as its next address.

A way of ~~the~~ traversing is required.

func Traverse()

~~current node = next node~~

next node = &current node . next address

current node = next node :

previous node = current node :

A way of adding a new list item.

func Add List item (obj)

New Node = new List Item { My Object = obj } ;

New Node . Next Address = current node . Next address ;

current node . Next Address = New Node . Address ;

And a way of removing a list item

func remove()

~~current node = next node~~

Previous Node . Next Address = ~~current node~~ current node . Next Address

This is the basis of a linked list but for an ordered one, more code would be necessary to ensure insertion at right place

func Insert (val)

{ inserted = false; current node = head node;
~~while (!inserted)~~ while (!inserted)

~~if (val > current node->data)~~
~~current node->next = new Node(val);~~
~~current node = current node->next;~~

memory

If (check val greater than node())

{
}
}

Traverse();

Else

{

TraverseBackward();

~~TraverseBackward();~~

Add List Item();

}

Inserted = true;

}

}

8
/ 10

This would be the basis of a more appropriate insert for an ordered linked list. One whereby each element inserted is checked against each other element to establish where in the list it should be inserted.

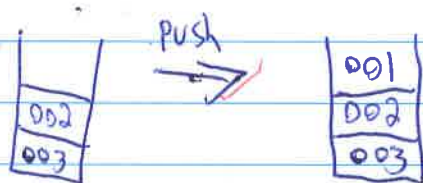
On average this should lead to half inserts checking few other values whilst others will check most. This leads to a complexity of $O(n^2)$.

2d-

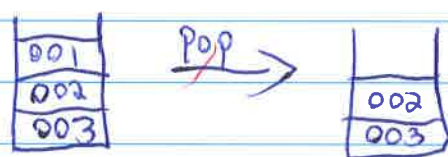
A stack is a Last In First Out data structure for containing a collection of objects.

To convert a linear linked list to a stack the tail of the list would have to become the first item on the stack and every other item would be added to the stack afterwards in reverse order if the list is ordered. *this constant needs to be broken*

The stack structure provides functionality to push a new item onto the stack.



And to pop an item off the top of the stack



Rather than needing to traverse a list by addressing it is now intrinsically ordered in the order things were added to the stack. The stack ensures the most recent addition is at the top.

To ~~reverse~~ ^{add and reorder} a stack, another stack can be used. Each value is popped off until the ~~new~~ right position has been found, these popped values however had been pushed onto a temp stack. Once the new value has been pushed onto the stack each value on

the temp stack can be popped and pushed back onto the main stack, this means that ordering is retained other than the newly added value.

could be better described.

Q2 =

$$\begin{array}{r} 11 \\ \hline 25 \end{array}$$

$$\begin{array}{r} 16 \\ \hline 25 \end{array}$$

3a.

pre order

NLR

SEACEERHRT ✓

~~In Order~~
~~NLR~~
~~EECAERHRT~~

In Order
LNR

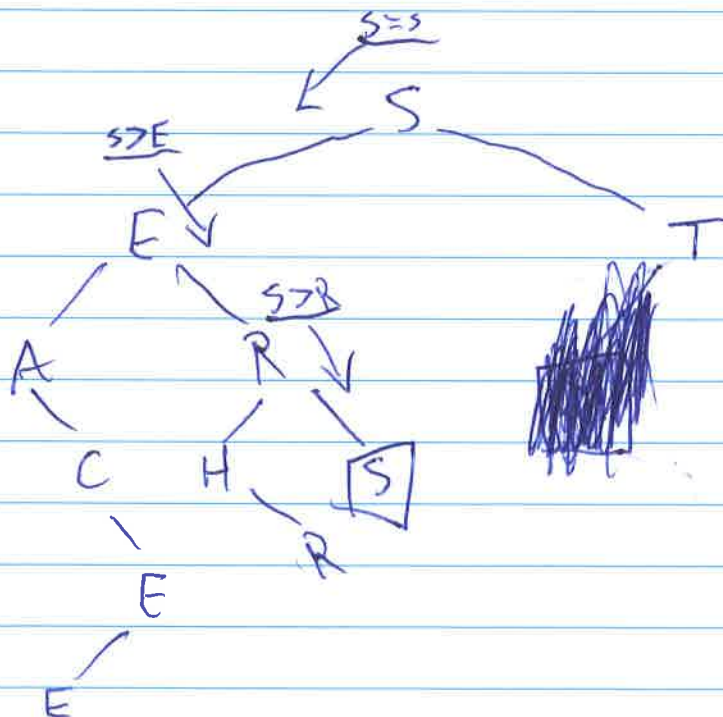
ACEEEHRRST ✓

Post order
LRN

EECARHRETS ✓

12/12

3B.



✓

12/12

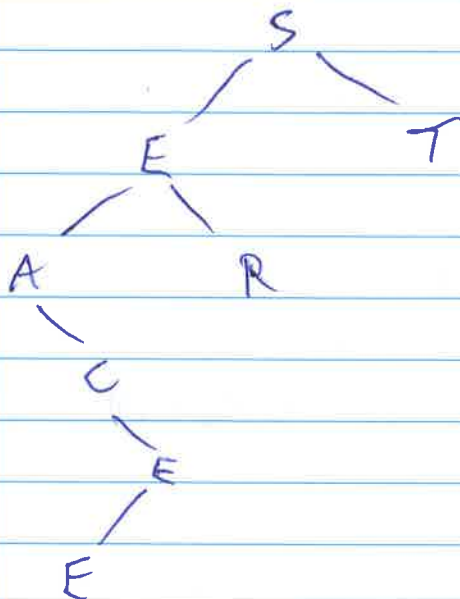
The presence of an E to the left of an E indicates that ~~more~~ a greater than is used for right hand branches not equal to. This meant that the S had to initially go left then continue right afterwards as whilst S is not greater than S, it is greater than everything less than S.

This binary tree facilitates searches well. However it is worth noting that it is a very unbalanced tree and given more ~~data~~ data that follows the precedent set so far, this could lead to issues.

but say both
enters
(w)
int)
There are two searches depth first search and breadth first search. Breadth first search reads left to right level by level. regardless of branch whereas depth first search would go left and drill down through the hierarchy until all its descendants have been evaluated before the right side would be evaluated in the same manner.

In the case of finding T depth first search would evaluate every value before it whereas it would be the third value evaluated using a breadth first search.

3C. To remove the H from the tree, its child nodes must be reassigned therefore the tree may change more than it would appear at first



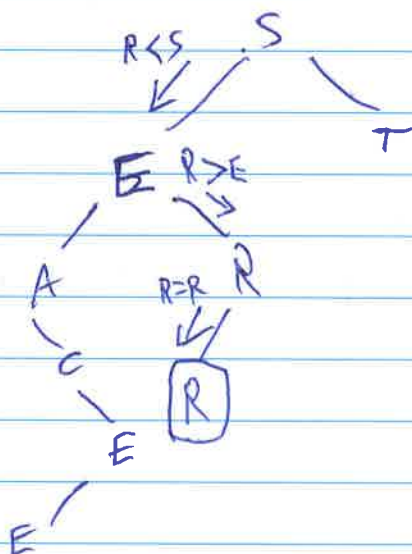
~~REMOVED~~

[R]

?

$2\frac{1}{2}$
5

This diagram is the tree with H removed and ~~REMOVED~~ its only child node R in memory ready to be placed back in the tree.



$Q3 = 20\frac{1}{2}$
25

$20\frac{1}{2}$
25

~~REMOVED~~ Due to R being a ~~rather~~ rather late in the alphabet but not after R, it has actually taken the position that once belonged to H. ~~REMOVED~~ ~~REMOVED~~

TYNNWCH A
RHWYGWCH

AR GYFER DEFNYYDD GWEINYDDOL YN UNIG, TYNNWCH Y STRIBYN TYLLOG I FFWRDD YN OFALUS

Rhowch yr holl wybodaeth y gofynnir amdani yn y blychau isod.

Rhaid i'r atebion fod mewn llawysgrïen **ddarlennadwy**. Gall yr Arholwyr naill ai anwyblyddu sgript annarllennadwy, neu drefnu iddi gael ei theipio a chael yr ymgeisydd i dalu.

Ysgriennwch eich atebion mewn inc.

Dechreuwch bob ateb ar dudalen newydd. Gwnewch unrhyw fras nodiadau yn y llyfr atebion a thynnu llinell drwyddynt.

Ni ddylid rhwygo unrhyw ran o'r llyfr hwn; rhaid ei gyflwyno yn gyfan.

Os defnyddir mwy nag un llyfr, rhwymwch hwy gyda'i gilydd gan ddefnyddio'r tagiau sydd ar gael oddi wrth y gwylt/wr/gwylt/wr.

Rhaid dychwelyd pob llyfr. Ni chaniateir mynd â llyfrau atebion, wedi eu defnyddio neu heb eu defnyddio, allan o'r lleoliad arholi o dan unrhyw amgylchiadau.

Rhif Myfyriwr	Teil y Papur Arholiad	Cod y Modwl	Dyddiad ac Amser yr Arholiad	Lleoliad yr Arholiad	Rhif Sedd
Nifer y Llyfrau Atebion a gyflwynwyd					

Rhestrwch ffrau'r cwestiynau a ymgestiwyd genych	
At Ddibenion y Swyddfa	Cyfanswm

(See the other side for English version)

A wneuch chi lenw'r rhan hon a'i SELIO cyn i'r arholiad ddechrau.

Cyfenw _____ (LLYTHRENNAU BREISION) Enw(au) cyntaf _____