

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	151713
2	Title of Examination Paper	Algorithms and Data Structures
3	Module Code	M5X01023
4	Date and Time of Examination	1/2/2018 ~~11:00~~ 11:18
5	Examination Venue	Rm MH312
6	Seat Number	

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

[illegible]

Please complete this section and SEAL before the examination starts.

EVANS
Surname

(Block Capitals) Forename(s) JONATHAN JEWAN

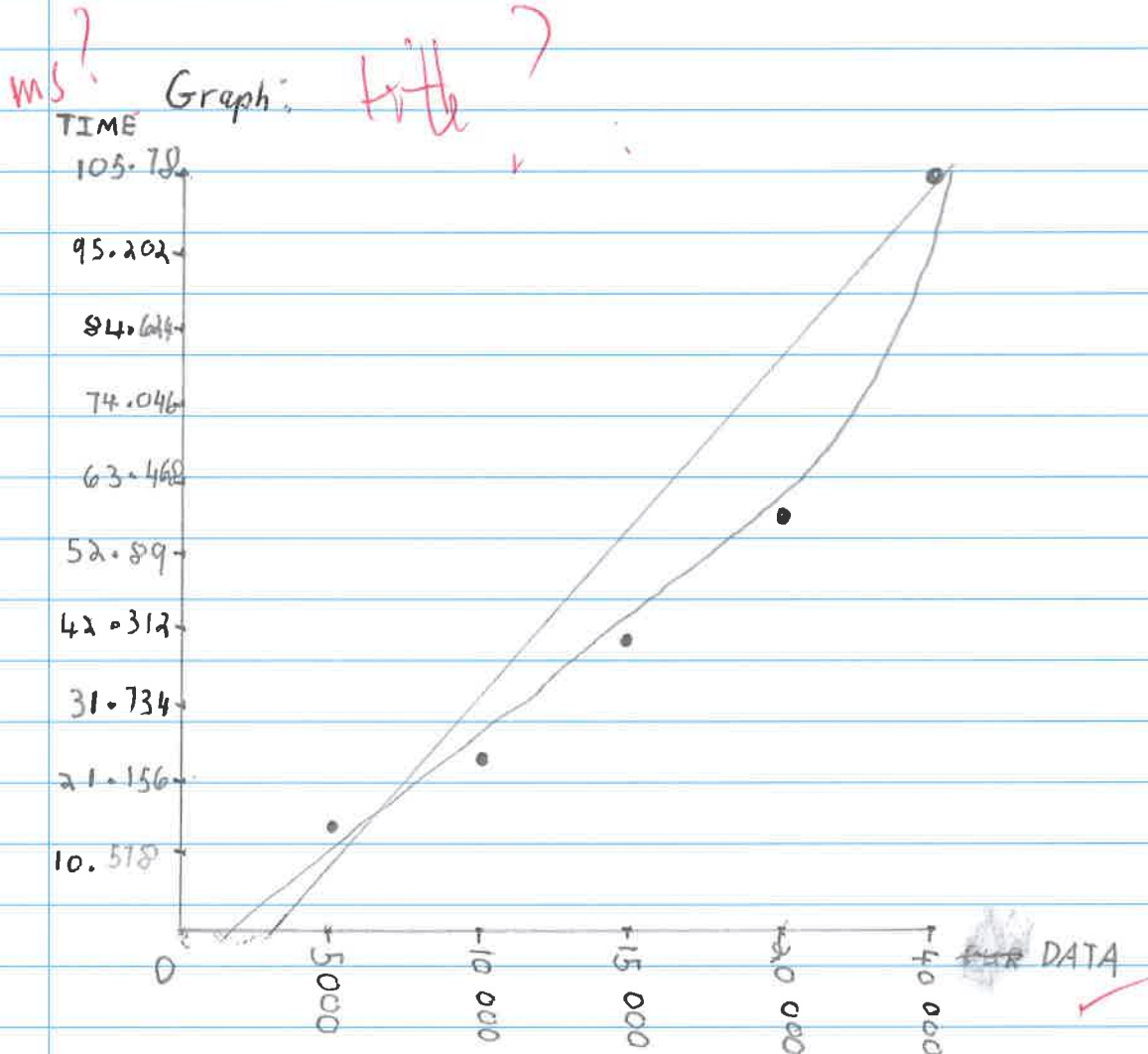
FOLD → Complete section, remove strip, fold as indicated and SEAL

PULL & TEAR

FOR ADMINISTRATIVE USE ONLY, CAREFULLY REMOVE PERFORATED STRIP

SECTION A

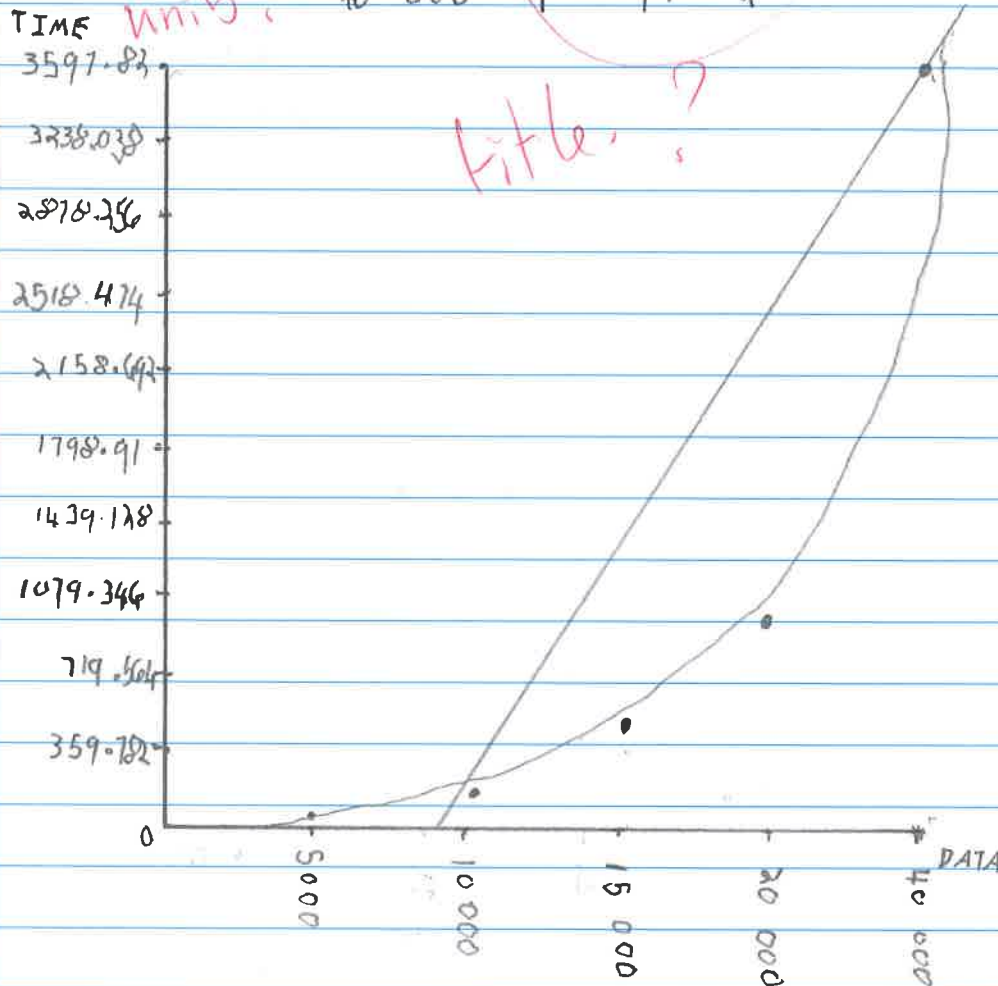
1. aFunc1 run times:	Data	Duration (milliseconds)
	5000	14.67
	10 000	25.64
	15 000	40.64
	20 000	56.08
	40 000	105.78



✓ The estimated complexity of this function is $O(n)$, linear time. The reason this is the case is because the function scales according to the data entered, primarily.

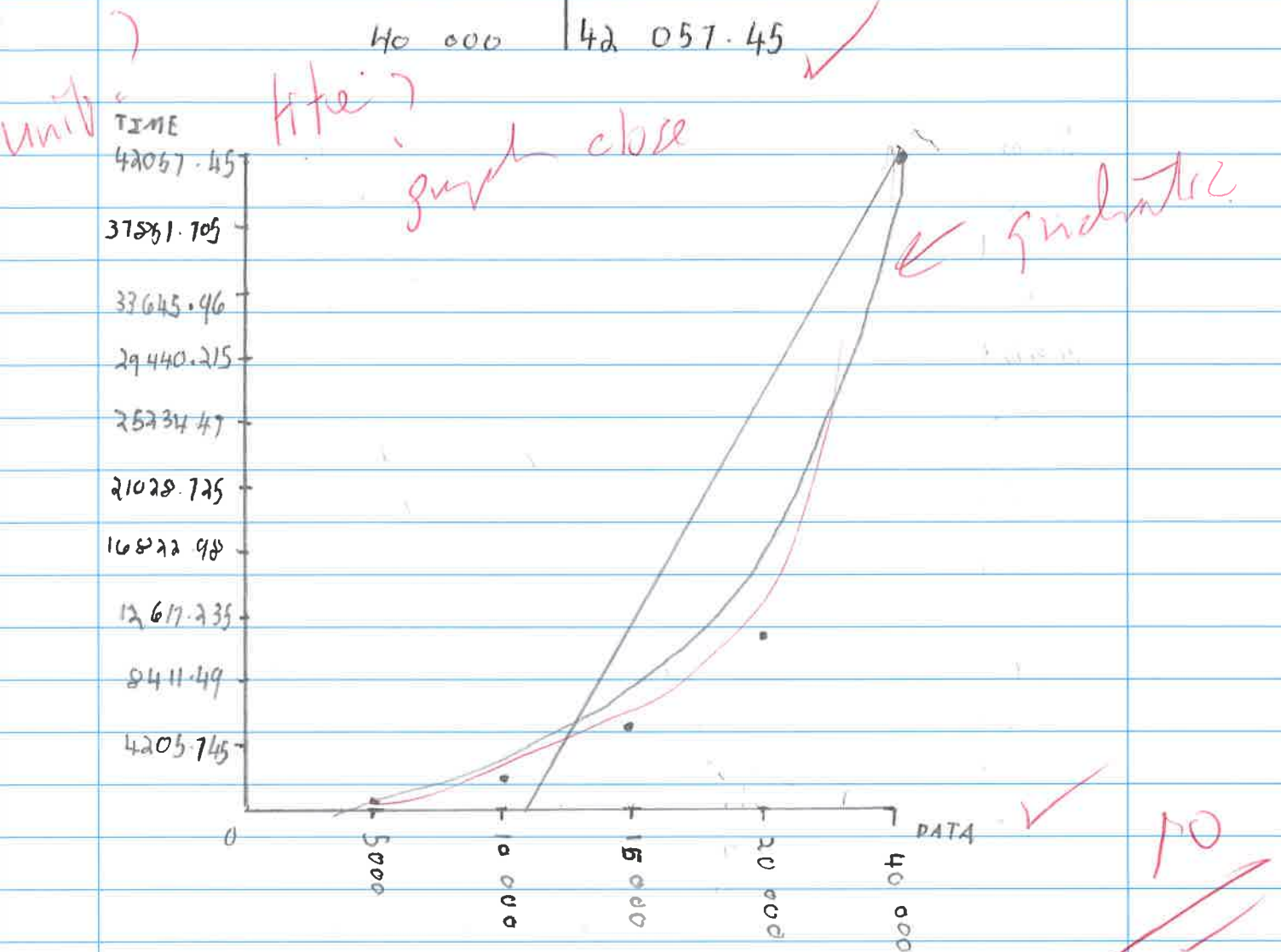
1.2 qFunc 2 runtimes :

Data	Duration (milliseconds)
5000	60.13
10000	~~56.23~~ 230.37
15000	515.61
20000	908.69
40000	3597.82



The estimated complexity of this function is $O(n^2)$, quadratic time. This is because the function ~~is within~~ uses a nested for loop, increasing the ~~time to~~ amount of operations by n^2 .

1.3 aFunc3 runtimes : Data	Duration (milliseconds)
5000	664.13
10000	2640.78
15000	5912.17
20000	10568.84
40000	42057.45



The estimated complexity of this function is $O(n^3)$, cubic time. This is because the function uses 3 levels of for loops.

10
25

10/25

SECTION B

2)

a) During program development, static memory is allocated at compile time. This is a ~~set~~ ^{fixed} amount.

Dynamic memory is allocated memory from the operating system's ^{heap} memory. The memory allocated must be created within program code. The programmer must make sure to free any allocated memory to ~~prevent~~ ^{prevent} heap corruption.

Dynamic memory is allocated during a ^{run} of a program, not at compile time, like static memory.

b) A Linearly linked list is an abstract data type created by the programmer. It allows the programmer to create a list ^{of data}, add to it, delete from it.

Search?

It consists of nodes. There are 3 main kinds of nodes: head nodes, List node, and a tail node.

Head node - This contains a pointer to the first item of the list.

List node - contains a pointer to the next item in a list, along with a unit of data.

Tail node - This terminates the list. It contains nothing, and points to NULL.

Can these hold data too??

Essential operations of a linked list:

- Add - Adds a node. This must come before the tail node. * It can be placed wherever desired.
 - To append to end - append after ~~after~~ the last ~~an~~ list node
 - To add between nodes the pointer of a previous node must point to the current one, and the current node must point to the succeeding node.

Reasons for inclusion:

- Useful when the programmer needs to create a list ~~with~~ with no set bounds,
✓ i.e.: a phone book.
- ✓ useful for managing collisions when used in hashing.

delete, search?

3/5

c). Ordered linked list pseudo code for addNode(node):

~~AddNode (node) {~~

~~Check each node^[i], if it smaller than \$node~~

~~IF TRUE?~~

~~Change node's pointer to \$node~~

~~Change \$node's pointer to node's pointer~~

~~ELSE~~

~~if~~

~~addNode (\$node) {~~

~~VARIABLE i = 0~~

~~WHILE~~

~~check node[i] > \$node~~

~~do: i++~~

~~ENDWHILE~~

~~VARIABLE tmp = node[i]~~

~~node[i] = point to \$node~~

~~node[i+1] = tmp~~

~~\$node = point to node[i+1]~~

```

addNode ($node) {
    var i = 0
    WHILE node[i] < $node
        do :
            i++
    ENDWHILE

```

```

    VAR tmp = node[i]
    Change node[i] pointer to $node
    change node[i+1] to VAR tmp = node[i+1]
    Change $node pointer to node[i+1]
}

```

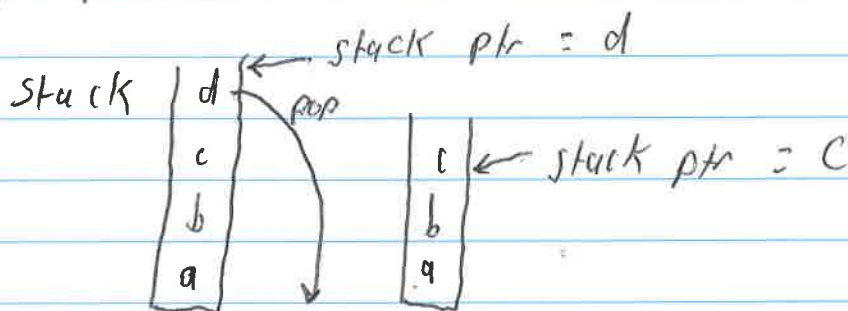
not sure what you mean - a little vague

The time complexity for this kind of insertion is $O(n)$, linear time. This is because the size of every node must be compared in the worst case.

average case is still $O(n)$

d). A linearly linked list can be modified to implement a stack by introducing a stack pointer that will keep track of the previously added node. As a result, the longest staying item of a list can be removed by changing the stack pointer to the ~~prev~~ next item.

Example :



stack pointer is fixed

This kind of storage is known as LIFO (Last-in, first-out).

$$\frac{16\frac{1}{2}}{25}$$

3).

Traversal:

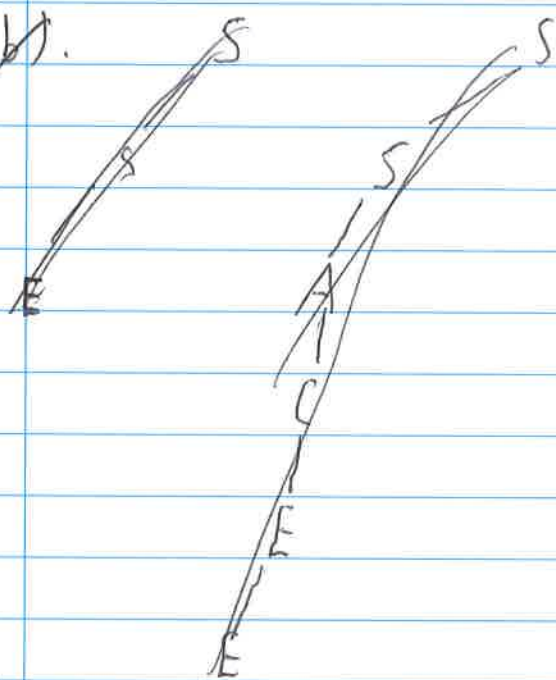
a) pre-order: S E C E E R H R T

post-order: ~~E E C A R H R E T S~~ E E C A R H R E T S

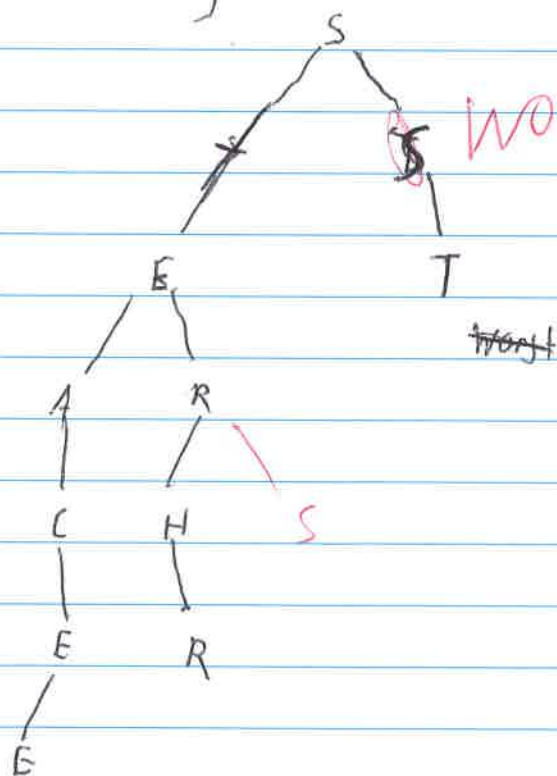
In-order: E E C A E R H R S T

9/2

b).



b). Inserting 'S'



Procedure:

An item is added to the tree by comparing the item not alphabetically.

Nodes that are positioned less, are attached to the left edge of a node.

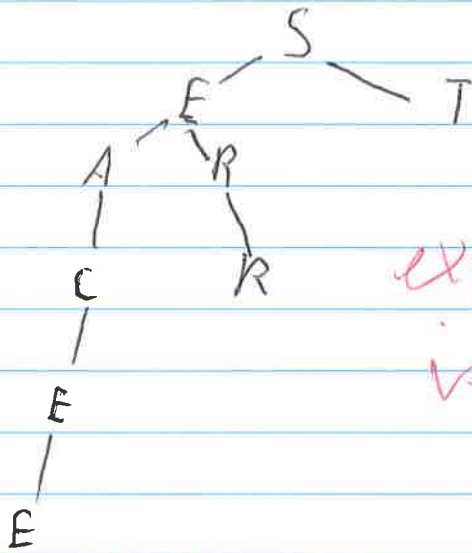
Nodes that are positioned higher, are attached to the right edge of the node.

8

The ~~worst~~ ^{best} case running time is $O(\log N)$ as the problem is divided.

The ~~best~~ ^{worst} case is $O(N)$, assuming the insert happens immediately.

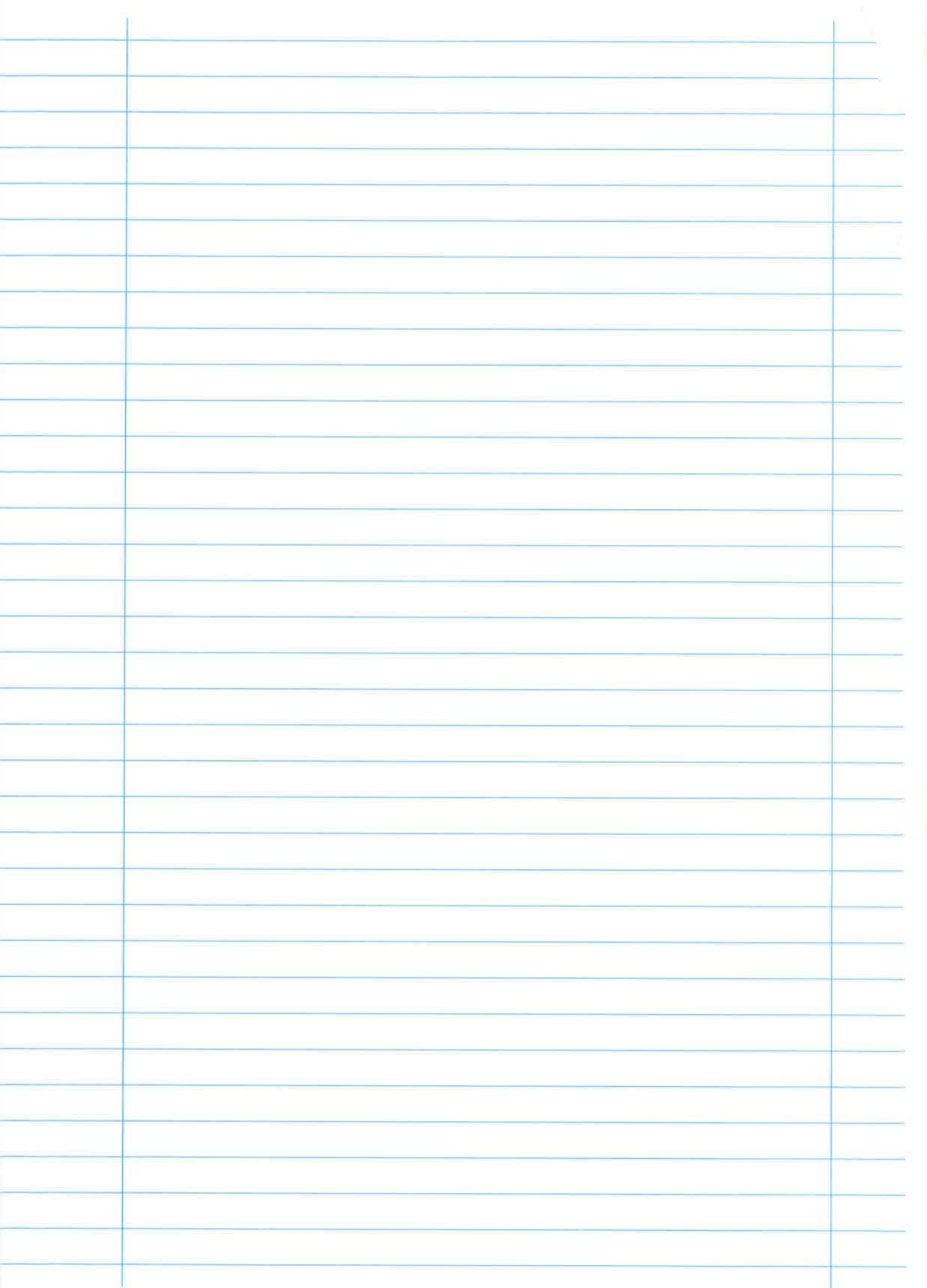
c). 'H' is deleted by removing the item and reattaching 'R' to 'R'. ok



explain how this is done in more detail

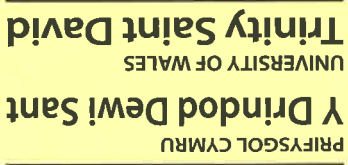
$$QJ = \frac{12^{1/2}}{25}$$

$$\frac{12^{1/2}}{25}$$



TYNNWCH A
RHWYGWCH

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



24 Tudalen

1. Rhowch yr holl wybodaeth y gofynnir amdant yn y blychau isod.
2. Rhowch eich enw yn y lle a ddarperir i'r dde a seliwch i guddio eich enw.
3. Rhaid i'r atebion fod mewn llawysgrifen **ddarlennadwy**. Gall yr Arholwyr natli ai anwybyddu sgript annarlennadwy, neu drefnu iddi gael ei theipio a chael yr ymgeisydd i dalu.

4. Ysgriennwch eich atebion mewn inc.
5. Dechreuwch bob ateb ar dudalen newydd. Gwnewch unrhyw fras nodiadau yn y llyfr atebion a thymu llinell drwyddynt.
6. Ni ddylid rhwygo unrhyw ran o'r llyfr hwn; rhaid ei gyflwyno yn gyfan.
7. Os defnyddir mwy nag un llyfr, rhwymwch hwy gyda'i gilydd gan ddefnyddio'r tagiau sydd ar gael oddi wrth y gwylwyr/gwylwyr.

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

1	Rhif Myfyrwr	
2	Teitl y Papur Arholiad	
3	Cod y Modwl	
4	Dyddiad ac Amser yr Arholiad	
5	Lleoliad yr Arholiad	
6	Rhif Sedd	
7	Nifer y Llyfrau Atebion a gyflwynwyd	

Rhestrwech rffau'r cwestiynau a ymgeistwyd gennyh	
At Dibenion y Swyddfa	Cyfanswm

(See the other side for English version)

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

Cyfenw

(LLYTHRENNAU BREISION) Enw(au) cyntaf

PLYGWCH

Lenwch y
rhan hon,
tynwch y
strilyn,
plygwch fel
a ddangosir
a SELWCH