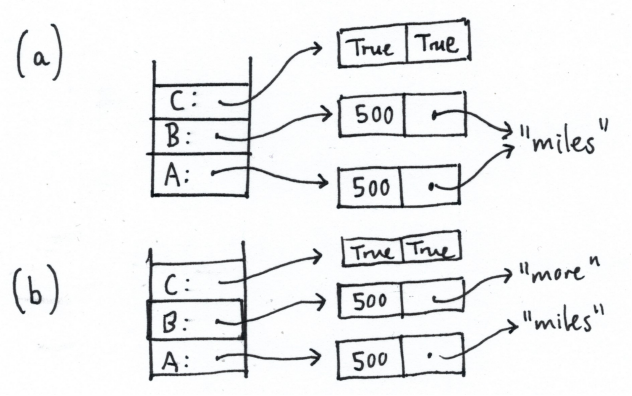
Module Title: Informatics 2 – Introduction to Algorithms and Data Structures
Exam Diet: April 2022
Brief notes on answers:
PART A:

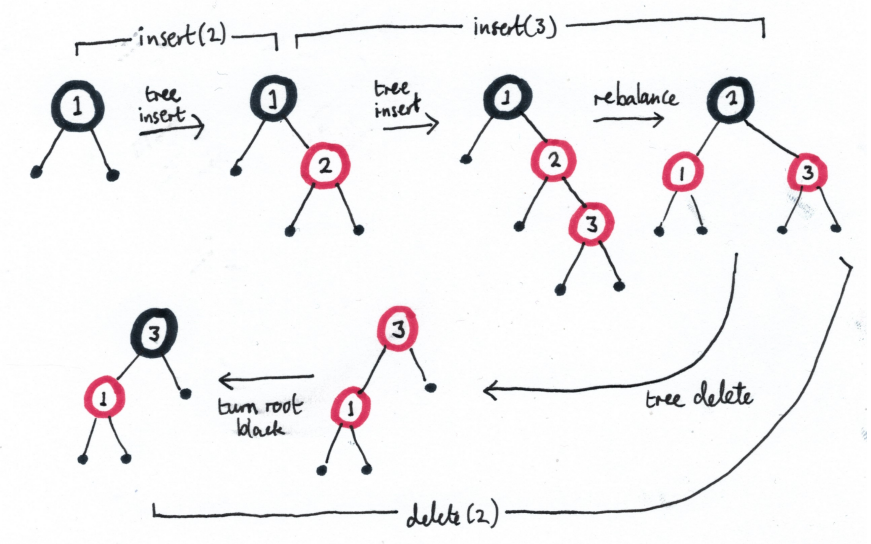
1. The required diagrams are:



   In (a), 1 mark for the stack entries, 2 marks each for heap data for A,B,C.

   In (b), 1 mark each for A,B,C.

2. The expected steps are:

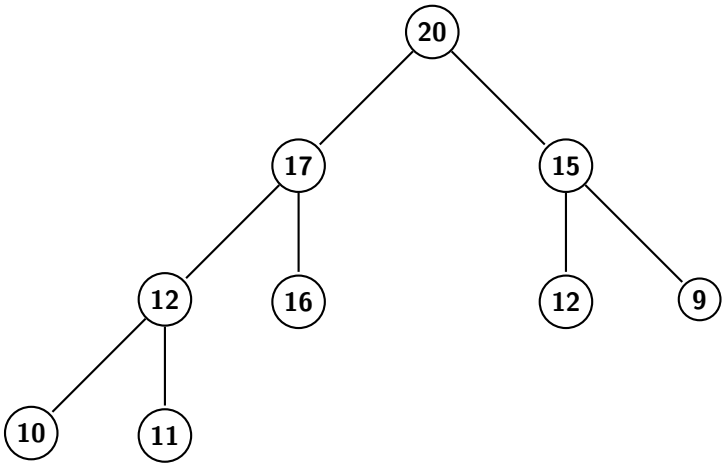


   insert(2): 2 marks.

   insert(3): 4 marks. Roughly 1 for the tree insertion, 3 for the rebalancing.

   delete(2): 4 marks. Roughly 3 for the tree deletion, 1 for turning the root black. The mirror-image solution is equally acceptable here.

3. (a) After the 4 operations have been carried out, the Heap will have the following shape:



20	17	15	12	16	12	9	10	11	-

Marking: 3 marks for the right answer (either in Heap form or array form), and 3 marks for showing some details of the 3 intermediate steps.

(b) In the case of Heap-Extract-Max, the method takes the *lowest/last item* in the Heap and copies that to the root, then calls Max-Heapify at the root. In the case where this copied item has the lowest key value 1, the Max-Heapify will continue making recursive calls down the Heap until it reaches another item with key 1. It is possible (for example in a tree where all items but the last one have key 2), that Max-Heapify might recurse for the entire height of the tree. Hence even in this restricted-key case, we have a lower bound of $\Omega(\lg(n))$ on the running-time of Heap-Extract-Max.

In the case of Max-Heap-Insert, the approach is to add the new item into the next free leaf on the bottom row of the Heap ($\Theta(1)$), and then consider "swapping-up" until the key k' of the parent is $\geq$ the key of the new item. Note that in some cases this may happen all the way to the root of the Heap (if the new item is the first with key 3, say), hence again we have worst-case running-time proportional to the height of the Heap, so $\Omega(\lg(n))$.

Marking: 1 mark for the height of the Heap being $\Theta(\lg(n))$, 1 mark each for explanations of how we could still have work proportional to height even in this restricted case, and 1 mark for the $O(\lg(n))$.

4. (a) (worked example, 6 marks)

Here are the initial rows of the two tables for the aligned words. Recall that in the a array, that a 0 indicates a match in the final column of the alignment, a 1 indicates a *substitution*, a 2 indicates an *insertion* (- matched against the final char of the prefix of "carnet") and a 3 indicates a *deletion*.

$$d = \begin{array}{c} \\ c \\ a \\ r \\ n \\ e \\ t \end{array} \begin{array}{c} \begin{array}{ccccccc} a & r & o & u & n & d & \end{array} \\ \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 1 & 2 & 3 & 4 & 5 & 6 \\ 3 & 2 & 1 & 2 & 3 & 4 & 5 \\ 4 & 3 & 2 & 2 & 3 & 3 & 4 \\ 5 & 4 & 3 & 3 & 3 & 4 & 4 \\ 6 & 5 & 4 & 4 & 4 & 4 & 5 \end{bmatrix} \end{array}, a = \begin{bmatrix} - & 2 & 2 & 2 & 2 & 2 & 2 \\ 3 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 \\ 3 & 0 & 1/2 & 1/2 & 1/2 & 1/2 & 1/2 \\ 3 & 1/3 & 0 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 1 & 1/2 & 0 & 2 \\ 3 & 3 & 3 & 1/3 & 1 & 1/2/3 & 1 \\ 3 & 3 & 3 & 1/3 & 1/3 & 1 & 1/2/3 \end{bmatrix}$$

marking: 2 of the marks goes for computing the d table accurately (ignore a slip or two), 2 marks go for the explanation of the final two rows, and 2 marks goes for the a-table.

(b) (worked example, 4 marks)

There are 6 different alignments to achieve the edit distance of 5. To see this, we note the bottom-right cell of a has 3 options - insert, delete, substitute.

If we follow the "delete" (3) option to the cell above, that shows a path which moves diagonally twice (1, then 0) and left twice (2, 2) and then diagonally again ... so the delete leads to a unique path.

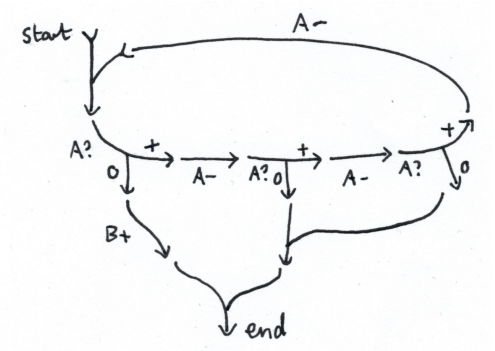
If we follow the "insert" (2) option to the cell at the left, this gives an entirely diagonal path (1, 1, 1, 0, 0) with no turns. Another unique explanation.

If we consider what happens with a substitution (1), this moves us 1 step up the diagonal, which again shows 3 options (1, 2, 3). Again, the delete (move up) and the insert (move left) options lead to unique paths (so 2 extra paths from these). The substitute (move up on diagonal) leads us to a cell showing 1/2, but when we follow either option the path becomes unique (so 2 extra paths from these).

So we get 6 paths total.

marking: Up to 4 marks, depending on the amount of detail. I will give 2 marks for any answer stating 6, even if students don't use the a table.

5. (a) Possible solution (minor variations acceptable):



3 marks for evidence of the right idea, 4 marks for correct details.

(b) Yes. We know from lectures that register machines are Turing complete — they can do anything that can be done on any classical model of computation, whi ch certainly includes primality testing. [1 mark for 'yes', 2 marks for the justification.]

PART B:

1. (a) (i) O and o. We have $f(n) = n \lg n$, and $\lg n = o(\sqrt{n})$ (standard fact). [2 marks for answer, 2 for justification.]
(ii) O, Ω, Θ. We have $f(n) = \sum_{k=1}^{n} \lg k \leq n \lg n$, which gives us O. Also $f(n) \geq \sum_{k=\lceil n/2 \rceil}^{n} \lg k \geq (n/2) \lg(n/2)$, and e.g. $\lg(n/2) \geq \lg(n)/2$ once $n \geq 4$, so $f(n) \geq (n/4) \lg n$, which gives us Ω and hence Θ. [Again 2 for answer, 4 for justification. This is harder, though they've seen something quite similar on a tutorial sheet.]

   (b) (i) A takes at worst $7n \lg n \lg \lg n$ steps, and B takes at best $105n \lg n$ steps. So A will beat B as long as $\lg \lg n < 105/7 = 15$. So take $N_1 = 2^{2^{15}}$. This is clearly the tightest value possible, as it's consistent with the data that A and B indeed take these times. (ii) A takes at best $6n \lg n \lg \lg n$ steps, and B takes at worst $120n \lg n$ steps. So B will beat A as long as $\lg \lg n > 120/6 = 20$. So take $N_2 = 2^{2^{20}}$.
   [1 mark for an idea on the right lines. 1 mark each for correct values of N_1 and N_2. 3 marks in total for the justifications.]
   Note: although the constants chosen here are fictitious, Algorithm B is indeed an example of a so-called 'galactic algorithm'.

   (c) We need to show

$$\forall c > 0.\ \exists N.\ \forall n \geq N.\ n \lg n < cn \lg n \lg \lg n$$

   But $n \lg n > 0$ for all $n > 1$, so this amounts to

$$\forall c > 0.\ \exists N.\ \forall n \geq N.\ 1 < c \lg \lg n$$

   So suppose given $c > 0$. Take any $N > 2^{2^{1/c}}$. Then for any $n \geq N$, we have $n > 2^{2^{1/c}}$, so $\lg \lg n > 1/c$, so $1 < c \lg \lg n$ as required.
   [3 marks for a correct statement of what we have to prove. 3 marks for 'given c, take $N > 2^{2^{1/c}}$ (may deduct 1 if they have $=$ here). 3 marks for the remaining details, being fairly strict.]

2. (a) The course of computation is:

Operation	Input left	Stack state
	com opt \$	Shell
Lookup com, Shell	com opt \$	com Args
Match com	opt \$	Args
Lookup opt, Args	opt \$	Opts Files
Lookup opt, Opts	opt \$	opt Opts Files
Match opt	\$	Opts Files
Lookup \$, Opts	\$	Files
Lookup \$, Files	\$	stack empties: success!

   [Roughly, 2 marks for a table of the right form, then 1 mark per line.]

   (b) Trying to parse 'com com':

Operation	Input left	Stack state
	com com \$	Shell
Lookup com, Shell	com com \$	com Args
Match com	com \$	Args
Lookup com, Args	opt \$	Blank table entry at (com,Args)

[Roughly, 1 mark per line, and 1 mark for the nature of the failure.]

(c) This could not happen in this case. The only table entries that can introduce terminals in the stack are those for c om Args, opt Opts, file Files, and each of these is triggered only when the relevant terminal (com, opt, file) has just been seen. The terminal will therefore be matched immediately after the rule in question ha s been applied.

[1 mark for 'no', 1 mark for seeing why not, 2 marks for quality of explanation.]

(d) The grammar is not ambiguous. Given any string s derivable from S, it is clear from s what production we must apply first. If $s = \epsilon$ then there is nothing mor e to do; otherwise we effectively have a shorter string s' to derive from S and may c ontinue in the same way.

[1 mark for 'no', 2 marks for explanation, not being too strict.]

However, the grammar is not LL(1). Informally, without lookahead there is no way to tell when we have reached the middle of the string, at which point we would wish to expand the predicted S to ϵ and start matching the remaining characters against those on the stack.

[1 mark for 'no', 2 marks for seeing the reason.]

3. (a) (15 marks, worked example)

$$\begin{aligned}\Phi \;=\; & (x_1 \vee \bar{x_2} \vee \bar{x_3}) \wedge (\bar{x_1} \vee \bar{x_2} \vee x_3) \wedge (\bar{x_1} \vee x_2 \vee x_4) \wedge \\ & (x_1 \vee \bar{x_2} \vee \bar{x_4}) \wedge (x_1 \vee x_3 \vee x_4) \wedge (\bar{x_1} \vee \bar{x_3} \vee x_4) \wedge \\ & (x_2 \vee x_3 \vee x_4) \wedge (\bar{x_2} \vee x_3 \vee \bar{x_4}) \wedge (x_2 \vee x_3 \vee \bar{x_4}).\end{aligned}$$

solution: We are to consider the variables in order of index, and we note first that x_1 appears as a positive literal in 3 clauses, with $\bar{x_1}$ also appearing in 3 clauses. Hence the conditional expectation of $x_1 \leftarrow 1$ is identical to that of $x_1 \leftarrow 0$, hence we choose $x_i \leftarrow 0$, and the resulting formula is as follows:

$$\begin{aligned}\Phi \;=\; & (\cancel{x_1} \vee \bar{x_2} \vee \bar{x_3}) \wedge (\bar{x_1} \vee \bar{x_2} \vee x_3) \wedge (\bar{x_1} \vee x_2 \vee x_4) \wedge \\ & (\cancel{x_1} \vee \bar{x_2} \vee \bar{x_4}) \wedge (\cancel{x_1} \vee x_3 \vee x_4) \wedge (\bar{x_1} \vee \bar{x_3} \vee x_4) \wedge \\ & (x_2 \vee x_3 \vee x_4) \wedge (\bar{x_2} \vee x_3 \vee \bar{x_4}) \wedge (x_2 \vee x_3 \vee \bar{x_4}).\end{aligned}$$

The conditional expectation of this reduced formula (over choices for x_2, x_3, x_4) is then $3 + 3\frac{3}{4} + 3\frac{7}{8}$, which is $7 + \frac{7}{8}$.

Next we compare $x_2 \leftarrow 0$ against $x_2 \leftarrow 1$. We see that in the "not satisfied yet" clauses, $\bar{x_2}$ appears in 2 length-2 clauses and 1 length-3 clause, while the positive literal x_2 appears in 2 length-3 clauses.

The conditional expectation of setting $x_2 \leftarrow 0$ is then $6 + 3\frac{3}{4} = 8 + \frac{1}{4}$.

The conditional expectation of $x_2 \leftarrow 1$ is less than this (as the average was $7+\frac{7}{8}$).

Hence we set $x_2 \leftarrow 0$ and our reduced formula is:

$$\begin{aligned}\Phi \;=\; & (\cancel{x_1} \vee \bar{x_2} \vee \bar{x_3}) \wedge (\bar{x_1} \vee \bar{x_2} \vee x_3) \wedge (\bar{x_1} \vee x_2 \vee x_4) \wedge \\ & (\cancel{x_1} \vee \bar{x_2} \vee \bar{x_4}) \wedge (\cancel{x_1} \vee x_3 \vee x_4) \wedge (\bar{x_1} \vee \bar{x_3} \vee x_4) \wedge \\ & (\cancel{x_2} \vee x_3 \vee x_4) \wedge (\bar{x_2} \vee x_3 \vee \bar{x_4}) \wedge (\cancel{x_2} \vee x_3 \vee \bar{x_4}).\end{aligned}$$

Next we must consider x_3 and we note that x_3 appears as a positive literal in *all* three remaining clauses. Therefore, we definitely know that the conditional expectation with $x_3 \leftarrow 1$ is $\geq$ than setting $x_3 \leftarrow 0$.

Hence we set $x_3 \leftarrow 1$, satisfying all clauses. We can set x_4 to either bit value. Hence the assignment $x_1 \leftarrow 0, x_2 \leftarrow 0, x_3 \leftarrow 1$ and x_4 either.

marking: 4 marks each for an accurate treatment of x_1, of x_2 and of x_3, giving details of the workings in each case. The final 3 marks for wrapping things up and noting x_4 may have either value.

(b) (2+8 marks, new thinking)

(i). In the case of a 4-CNF formula, each individual clause has the option to be satisfied by any of 4 independent literals. On a uniform random assignment, the chance that a specific clause is unsatisfied is only $1 - \frac{15}{16}$, with probability $\frac{15}{16}$ of being satisfied. Given there are m clauses, the expected number of clauses is $\frac{15}{16}m$.
marking: 2 marks for the right answer.

(ii). We are asked to consider the effect of setting $x_i \leftarrow 1$ as opposed to $x_i \leftarrow 0$, given the counts in the question. For any clause already T/F, or any remaining clause not-involving x_i, the expectation is unaffected by x_i's assigned value. We only need focus on the clauses that remain that involve x_i or $\bar{x_i}$.
If we assign $x_i \leftarrow 0$, then every clause containing $\bar{x_i}$ becomes satisfied - all $k_1^- + k_2^- + k_3^- + k_4^-$ of them. On the other hand, the clauses containing x_i have their size reduced by 1 - so we end up with k_2^+ clauses of length 1 (each with probability $\frac{1}{2}$ of being satisfied), k_3^+ clauses of length 2 (each with probability $\frac{3}{4}$ of being satisfied), and k_4^+ clauses of length 3 (each with probability $\frac{7}{8}$ of being satisfied).
Hence the overall contribution of $x_i \leftarrow 0$ to the expectation will be $k_1^- + k_2^- + k_3^- + k_4^- + \frac{1}{2}k_2^+ + \frac{3}{4}k_3^+ + \frac{7}{8}k_4^+$.
On the other hand, the overall contribution of $x_i \leftarrow 1$ to the expectation will be $k_1^+ + k_2^+ + k_3^+ + k_4^+ + \frac{1}{2}k_2^- + \frac{3}{4}k_3^- + \frac{7}{8}k_4^-$.
Then the conditional expectation for $x_i \leftarrow 1$ is greater than or equal than for $x_i \leftarrow 0 \Leftrightarrow$

$$\begin{aligned} k_1^+ + k_2^+ + k_3^+ + k_4^+ + \tfrac{1}{2}k_2^- + \tfrac{3}{4}k_3^- + \tfrac{7}{8}k_4^- &\geq k_1^- + k_2^- + k_3^- + k_4^- + \tfrac{1}{2}k_2^+ + \tfrac{3}{4}k_3^+ + \tfrac{7}{8}k_4^+ \\ k_1^+ + \tfrac{1}{2}k_2^+ \tfrac{1}{4} + k_3^+ + \tfrac{1}{8}k_4^+ &\geq k_1^- + \tfrac{1}{2}k_2^- + \tfrac{1}{4}k_3^- + \tfrac{1}{8}k_4^-, \end{aligned}$$

and this linear inequality is in the form requested.
marking: 3 marks going for the discussion of what happens after $x_i \leftarrow 0$ (or setting to 1), with reference to the size of the clauses and the consequent "expected" values, 2 marks for getting the overall expectation correct for each case, 3 marks for bringing it into the correct form.