

No part of the candidate's evidence in this exemplar material may be presented in an external assessment for the purpose of gaining an NZQA qualification or award.

## MERIT EXEMPLAR 2022



NEW ZEALAND QUALIFICATIONS AUTHORITY  
MANA TOHU MĀTAURANGA O AOTEAROA

QUALIFY FOR THE FUTURE WORLD  
KIA NOHO TAKATŪ KI TŌ ĀMUA AO!

# 3

COMMON ASSESSMENT TASK

### Level 3 Digital Technologies and Hangarau Matihiko 2022

#### 91908 Analyse an area of computer science

Credits: Three

Achievement Criteria		
Achievement	Achievement with Merit	Achievement with Excellence
Analyse an area of computer science.	Analyse, in depth, an area of computer science.	Critically analyse an area of computer science.

Type your School Code and 9-digit National Student Number (NSN) into the space below. (If your NSN has 10 digits, omit the leading zero.) It should look like “123-123456789-91908”.

-91908

There are three questions in this document. **Choose ONE question to answer.**

Make sure you have the PDF of Resource Booklet 91908R. This contains resources for Questions Two and Three.

You should aim to write **800–1500 words** in total.

Your answers should be presented in 12pt Times New Roman font, within the expanding text boxes, and may include only information you produce during this assessment session. Internet access is not permitted.

**Save your finished work as a PDF file** with the file name used in the header at the top of this page (“SchoolCode-YourNSN-91908.pdf”).

By saving your work at the end of the examination, you are declaring that this work is your own. NZQA may sample your work to ensure this is the case.

## INSTRUCTIONS

There are three questions in this assessment, on the topics of:

- Formal languages ([page 3](#))
- Computer graphics ([page 13](#))
- Computer vision ([page 19](#)).

**Choose ONE question to answer.**

Questions Two and Three require you to refer to the separate resource booklet.

Read all parts of your chosen question before you begin.

## ***EITHER:* QUESTION ONE: Formal languages**

### (a) Deterministic finite automata

A deterministic finite automaton (DFA) can be described by a five-element tuple  $(Q, \Sigma, \delta, q_0, F)$  where:

- $Q$  is a finite set of states
- $\Sigma$  (sigma) is a finite, non-empty input alphabet
- $\delta$  (delta) is a series of transition functions
- $q_0$  is the starting state
- $F$  is the set of accepting states.

Figure 1 shows a deterministic finite automaton.

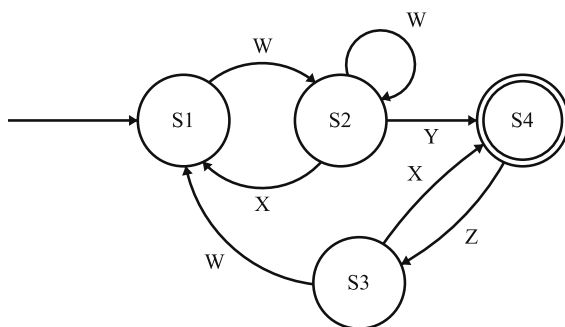


Figure 1.

(i) Complete the following for the DFA in Figure 1.

$Q =$	S1, S2, S3, S4
$\Sigma =$	W, X, Y, Z

$\delta =$	Current state	Input symbol	Next state
	S1	W	S2
	S2	W	S2
	S2	X	S1
	S2	Y	S4
	S3	W	S1
	S3	X	S4
	S4	Z	S3

$q_0 =$	S1
$F =$	S4

(ii) Which of the following strings are not accepted by the DFA in Figure 1?

- (1) WXWYZWWY
- (2) WXWYZX
- (3) WWWWWY
- (4) WWWWYZ

4

(iii) Explain how you came to this conclusion.

Strings 1-3 is accepted because after the string is fully inputted the final state will be S4 the accepting state.

When string 4 is inputted the final state will instead be S3 which is a non-accepting state therefore string 4 will not be accepted by the FSA.

- (b) The following table shows syntax that is sometimes used for regular expressions:

Expression	Description
[a-z]	Any single character in the range A–Z
[0–9]	Any single number in the range 0–9
+	One or more repetitions of the preceding element
*	Zero or more repetitions of the preceding element
?	Zero or one of the preceding element
.	Any character
\d	Any digit
[abc]	Only a, b, or c
[^abc]	Not a, b, or c
\w	Any alphanumeric character
\W	Any non-alphanumeric character

Consider the regular expression **[CFR]an**

- (i) Which words from the following list does the expression describe?

Can, Dan, Fan, Man, Pan, Ran

Can, Fan and Ran

- (ii) Explain how you came to this conclusion.

This regular expression states that the first character can only be C, F or R which is indicated by the [] around the CFR characters. The next two characters must be a, n this set of conditions is fulfilled by only Can, Fan and Ran. All given strings end in an however only these 3 starts with C, F or R which is why they are the only ones found by the Regex.

- (iii) What regular expression could you write that would find all of the words in the following list?

babble, bebble, bibble, bobble, bubble

b[aeiou]bble

- (iv) Explain how you came to this conclusion.

The first character in all these strings is b so my regular expression starts with a b. The second character is different for every string so I used the [] meta characters to indicate that the second character must be one of a, e, i, o or u finally all given strings end with bble so that is what my regular expression ends with. This regular expression will find all words in the above list as my regular expression matches the format of all the items on the list.

- (c) Figure 2 shows the production rules of a context-free grammar, and Figure 3 shows productions which have been applied to non-terminals from left to right in this order.

$E \rightarrow N$   
 $E \rightarrow E + E$   
 $E \rightarrow E * E$   
 $E \rightarrow -E$   
 $E \rightarrow (E)$   
 $N \rightarrow 0-9$

Figure 2.

$E \rightarrow E * E$   
 $E \rightarrow (E)$   
 $E \rightarrow E + E$   
 $E \rightarrow N$   
 $E \rightarrow N$   
 $E \rightarrow N$   
 $N \rightarrow 7$   
 $N \rightarrow 3$   
 $N \rightarrow 2$

Figure 3.

- (i) What are the non-terminal symbols used in the production rules in Figure 2?

E, N

- (ii) What are the terminal symbols used in the production rules in Figure 2?

\*, (, ), +, 0-9, -

- (iii) Give the expression built by the sequence in Figure 3, showing the result from applying each production.

Initial: E  
 Step 1  $E \rightarrow E * E$  result:  $E * E$   
 Step 2  $E \rightarrow (E)$  result:  $(E) * E$   
 Step 3  $E \rightarrow E + E$  result:  $(E + E) * E$   
 Step 4  $E \rightarrow N$  result:  $(N + E) * E$   
 Step 5  $E \rightarrow N$  result:  $(N + N) * E$   
 Step 6  $E \rightarrow N$  result:  $(N + N) * N$   
 Step 7  $N \rightarrow 7$  result:  $(7 + N) * N$   
 Step 8  $N \rightarrow 3$  result:  $(7 + 3) * N$   
 Step 9  $N \rightarrow 2$  result:  $(7 + 3) * 2$   
 Expression built by the production is  $(7 + 3) * 2$





- (iv) Explain what it would mean if there was no way to generate a particular expression using these productions. You may wish to illustrate this with an example that cannot be generated by the above grammar.

If there was no way to generate an expression using the production rules of the context free grammar given in figure 2 then the rules must not contain a way for the desired character to be produced. An example of an expression this context free grammar cannot produce would be  $(7+3)/2$  the given production rules provide no way to produce the  $/$  character. The production rule needed to allow this expression would be  $E \rightarrow E/E$  however as this is not provided you cannot make my expression. You could get around this problem by adding the rule above to the rules in figure 2.

- (d) If a language can be recognised by a finite state machine (FSM) it is said to be a regular language, and if there is a finite state machine that recognises a language, then that language must be a regular language.

Using examples to support your reasoning, explain what this statement means.

You might consider:

- the relative capabilities of formal languages that you have studied
- the limits of what they can do.

This statement means that any language which can be inputted into a finite state machine is a regular language. The purpose of an FSM is to determine if a given string complies with a series of rules. An FSM for the English language would have the purpose of checking to see if strings comply with the rules of the language. So, the first problem would be what are the rules of the English language, how to you codify what makes up a valid word, sentence or paragraph. A computer would struggle to understand why:

this is a valid sentence while; is valid.

sentence a valid this while, while this; is invalid.

It would of course be possible with sufficient rules to build an FSA for the English language the resulting complexity would be nearly infinite to the point were doing so can be considered impossible. Therefore, English is not considered to be a regular language.

This problem is caused by one of the major limitations of formal languages they are far better at determining if a string fits a valid format than if the string itself is valid. For example, Regex which is a regular language as it is made up of a series of metacharacters with very concrete non-contextual meanings could be validated by an FSA however it would not be able to determine if the Regex made sense. For example a Regex FSA could determine that the string `[A-Z][a-z]*` is a valid Regex statement as it follows all of Regexs syntax rules. While it could also determine that `[A-Z][a-z*` is invalid as it is missing the `]` metacharacter so it is violating the syntax rules of Regex. This FSA could not determine that is a nonsensical regex statement `^[A|A]` as it complies with all the Regex rules however it is asking that a character is either not A or A which is an entirely redundant statement as all characters would be valid inputs. This is a major limitation of formal languages.

A more significant example of this limitation would be if you tried to create a Regex statement to verify passwords. If you simply wanted to verify the format of a password we could use `^[A-Z][a-z]{5}[0-9]{4}@$. This statement will validate all passwords that start with a capital letter which is followed by 5 lower case letters then 4 numbers and finally an @ sign. If you wanted to verify the password however your Regex statement would need to contain every password in the database and a lot of | or metacharacters like: (Steven2000@)|(Adamdv2016@)...` This Regex would be impractically long. Therefore a major limitation of formal languages is how complicated they can become when checking two strings against each other they are much better at checking if a string matches a format.

- (e) It is possible to write a simple program that can recognise any string in the language **HH**, where a string is made up of a number of 0s followed by the same number of 1s.

Here is an example string, where six 0s are followed by six 1s:

000000111111  
                        
      H      H

Is it possible to create a finite state automata (FSA) for this language? Explain your answer by identifying the key problem related to regular languages (languages that use regular expressions and finite state automata). How do we get around this problem?

With the provided meta characters it would be possible to use regex to validate strings of HH although it would be infinitely long. The only rule for checking if a string of HH is valid is that it should start with a 0 end with a 1 and contain an equal number of 0s or 1s. This is a very simple rule however when you convert it into a regular language it becomes very problematic. You would need to write a rule for every number of 0s or 1s ie (01)|(0011)|(000111)... this is because Regex lacks the ability to go back and check if the same number of 0s and 1s have been entered so instead you have to define each combination as shown above. This makes this simple problem infinitely complex assuming there is no limit to the length of HH strings. To get around this problem you would need a more advance series of meta characters that would be able to simplify this problem. So, lets create a new metacharacter that could simplify this problem. It has two rules:

- 1) There must be at least 1 of the character in front of and behind it
- 2) There must be an equal number of the character in front of and behind it

We will call this character =. This would allow us to simply write 0=1 as our rule for HH. This = metacharacter however would be tricky to create as the two rules above would need to be written using other preexisting meta characters however creating this new character is a far more practical way of getting around this problem than attempting to account for every possible number of 0s/1s.

## Creating a game

Your friend is creating a simple computer game, but they need some help. They want to program the enemy sprites to chase the player's sprite, but aren't sure how to do so.

You have just finished a computer science topic on formal languages and suggest that an FSM could be a good starting point to solve their problem. You sketch the diagram in Figure 4 to get your friend started.

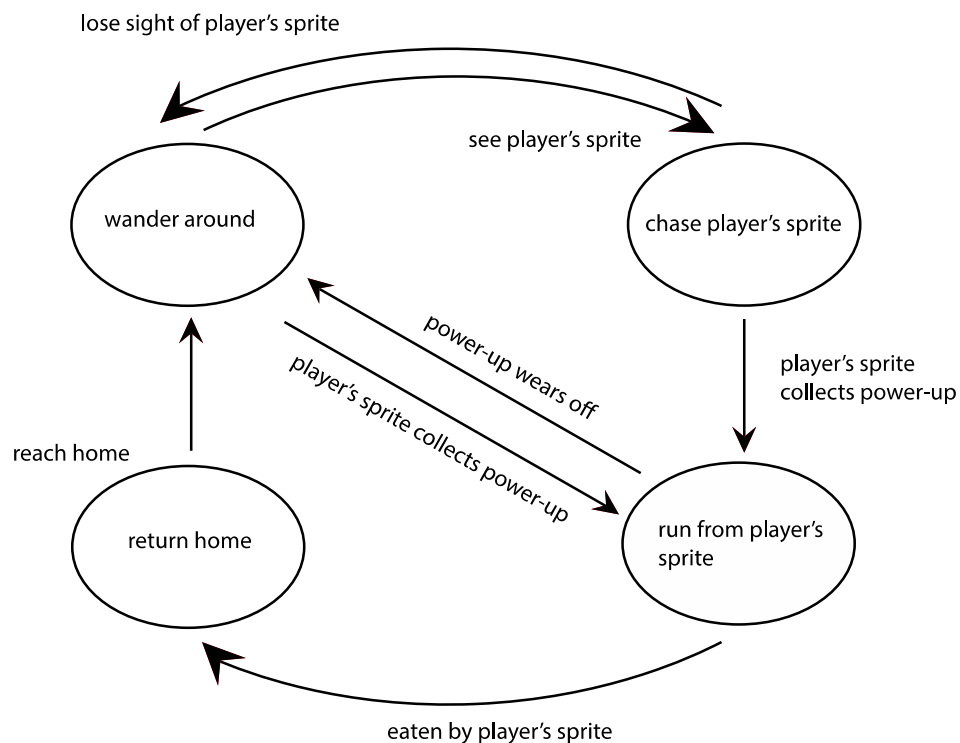



Figure 4. Behaviour of enemy sprite

- (f) Explain how your understanding of FSMs and formal languages could be used to help get your friend started. In your answer you may also discuss where else formal languages are used and how they affect people.

This FSA is a great way of illustrating the rules which the enemy sprite will follow. It has 4 possible behaviors/states and there are rules for under what conditions the sprite will change its behavior. This could help my friend understand that the sprite needs 4 sets of behaviors return home, wander around, run from player, chase player and then a series of rules for when to change its current behavior. This plan would help them figure out where they wanted to start when coding the behavior of the enemy sprite.

Formal languages are used all around us they are the basis for most coding languages, the worldwide web and even how the buttons on tv remotes work. In the case of the tv remote example the on/off button could be defined by a FSA. This FSA would have 3 states the first would be on the second would be confirm and the third would be off. The two inputs would be the on/off button itself and a cancel button. The TV would start off but when the on/off button



was pressed it would change to the on state. When the on/off button is pressed again it would change to the confirm state were a message asking the user if they were sure they wanted to turn the tv off would appear. Then if they pressed cancel the tv would return to the on state but if they pressed on/off again it instead go to the off state. These rules are defined by a formal language and are an example of how someone might interact with a formal language which we can visualize with an FSA on a day-to-day basis.

(g) Refer to the FSM in Figure 5.

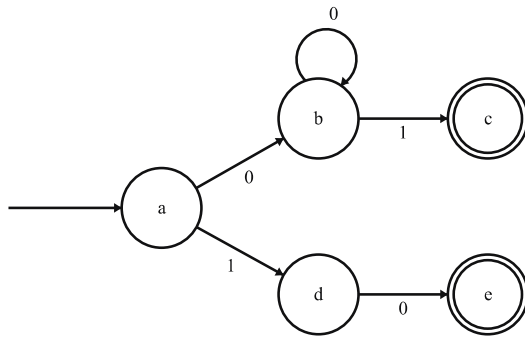


Figure 5.

(i) What does this machine accept?

Either any number of 0s followed by a 1 or 1 and then 0.

(ii) Explain what happens if the strings 111 or 1010 are applied.

The string 111 would not be accepted by the FSA there is no rule for what happens when 1 is entered and the active state is d so presumably it would remain on this state. As d is a nonaccepting state the input would be considered invalid by the FSA.

The string 1010 however would be accepted by the FSA as the 1, 0 input will make c the active state and the additional 1, 0 input will not change the active state as there are no rules for what happens under those circumstances so presumably state c would remain active causing the FSA to validate the result.

Depending on how the FSA operates however it is possible that entering an input that doesn't have a rule ie entering 1 on state d would simply cause the FSA to declare the string invalid as no rule is provided in this case both 111 and 1010 would be considered invalid inputs.



**This page has been deliberately left blank.**

**OR: QUESTION TWO: Computer graphics**

This question includes references to **Resources A, B, and C** on pages 2 and 3 of the resource booklet.

- (a) (i) What are matrix transformations used for in computer graphics?

- (ii) Why are matrix transformations used in computer graphics?



- (b) Translation, scaling, and rotation can all be performed on a single shape.

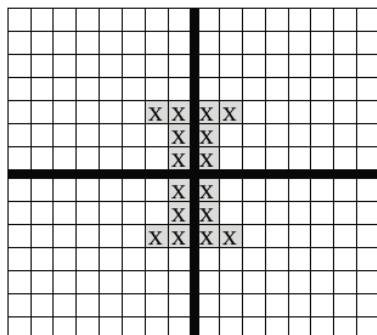
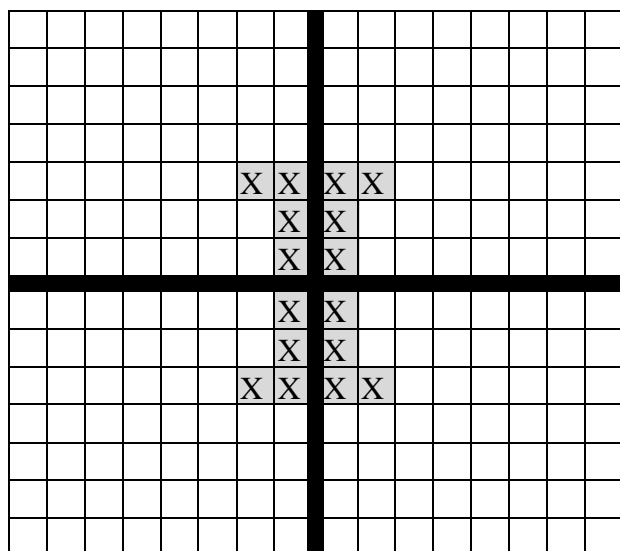


Figure 6.

- (i) Using the multiplication matrix  $\begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}$ , complete a 2D scaling effect on the shape in Figure 6 by filling in the new scaled shape with crosses (X) on the graph below.



- (ii) Using the matrix, explain how you knew where to position the point at (2,3) after scaling had been applied.

You may use the boxes below to support your answer.




- (c) (i) The mathematical formula for calculating the slope of a line is:

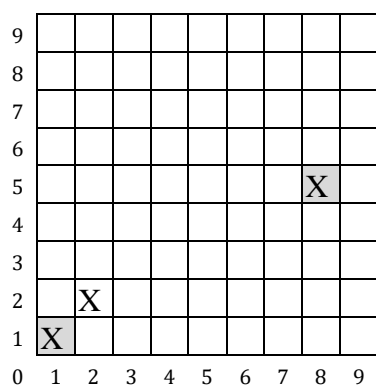
$y = mx + b$ , where  $m$  is the slope and  $b$  is the  $y$  intercept.

Explain why this formula works well for drawing lines on paper but does not work well for drawing lines on a computer screen. Refer to **Resources A and B** to support your answer.

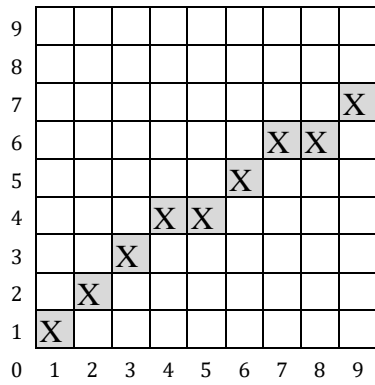
- (ii) Using the algorithm in **Resource B**, calculate the points that would be plotted in order to draw a line between (1,1) and (8,5).

Points plotted	P	x coordinate	y coordinate
1		1	1
2	-5	2	2
3			
4			
5			
6			
7			
8	1	8	5

- (iii) Fill in the pixels with the points you calculated in (ii). The first pixel has been done for you.



- (iv) Below is an example of a line drawn on a screen using the coordinates (1,1) and (9,7).



Explain the concept of anti-aliasing and demonstrate how it could be used to make this line appear smoother by adding crosses (X) in the table above.

### 3D Computer Graphics

Your friend has just bought a new computer and wants to play the latest high-resolution video games and do intensive 3D modelling. Their computer does not have a dedicated graphics card or GPU and they are complaining that the game lags and looks dull, and that rendering and creating 3D models is taking forever.

- (d) (i) How would you explain to your friend what is causing the problem? Refer to specific algorithms and computer science concepts in your answer.

- (ii) Explain how this same problem affects people in other scenarios where computer graphics are used.

- (iii) Explain how these problems are being solved.



**This page has been deliberately left blank.**

**OR: QUESTION THREE: Computer vision**

This question includes references to **Resources C to H** on pages 3 to 5 of the resource booklet.

- (a) (i) In relation to computer vision, explain the issue of noise. Refer to **Resources C and D** to support your answer.

- (ii) How are these issues addressed? Refer to **Resource D** to support your answer.

(b) You may refer to **Resources E to H** to support your answers for part (b).

(i) What is Canny edge detection used for?

(ii) Describe a simple way edge detection can be carried out.

(iii) What are the challenges involved in implementing accurate edge detection?

(iv) How are these challenges addressed?

- (c) Explain techniques that can be used to provide depth information in computer vision. What is needed to get good accuracy of depth?

- (d) (i) How do facial recognition systems work in order to recognise a face and identify it accurately?

- (ii) What is the difference between face detection and face recognition? What are some applications of each of these techniques?

- (iii) How can these applications impact on humans both positively and negatively?





Source (adapted): <https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e>

- (e) Referring to the statement above, explain how computer vision is changing and will continue to change. How could these changes have both positive and negative impact on humans? You may use an example you have studied this year to further support your answer.

## Acknowledgments

Material from the following sources has been adapted for use in this assessment:

"Theory of Computation"; Portland State University: Prof. Harry Porter

<https://research.ncl.ac.uk/game/mastersdegree/gametechnologies/previousinformation/artificialintelligence1finitestatemachines/2016%20Tutorial%208%20-%20Finite%20State%20Machines.pdf>

[https://en.wikipedia.org/wiki/Computer\\_graphics/](https://en.wikipedia.org/wiki/Computer_graphics/)

<https://towardsdatascience.com/everything-you-ever-wanted-to-know-about-computer-vision-heres-a-look-why-it-s-so-awesome-e8a58dfb641e>

## Merit Exemplar 2022

Subject	Digital Technologies and Hangarau Matihiko Level 3		Standard	91908	Total score	06
Q	Grade score	Annotation				
1	M6	<p>The candidate answered all components for Achieved with no significant errors. The candidate is able to briefly explain their responses and showed good understanding of the key aspects, the algorithms, and techniques.</p> <p>At Merit, the candidate shows good understanding of the concepts and uses their understanding to refer to examples provided as well as ones they've learnt independently, for example in the password verifications and Regex rules.</p> <p>The candidate was able to link to examples of other areas, but the response lacked the critical analysis required to move into the Excellence band. The candidate was able to understand and explain the Finite State Machine and Automata examples and was able to link them to everyday use. The candidate compared and contrasted examples from within the field of Formal Languages.</p> <p>The candidate did not expand their answers enough to show appropriate depth of response or insightful conclusions for the Excellence level step-up.</p>				