3

NZQA
NEW ZEALAND QUALIFICATIONS AUTHORITY
MANA TOHU MĀTAURANGA O AOTEAROA

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

COMMON ASSESSMENT TASK

Level 3 Digital Technologies and Hangarau Matihiko, 2019

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 header at the top of this page. (If your NSN has 10 digits, omit the leading zero.)

Answer all parts of the assessment task in this document.

Your answer should be presented in 12pt Arial font, within the expanding text boxes, and may only include information you produce during this examination session.

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

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 that this is the case.

YOU MUST HAND THIS BOOKLET TO THE SUPERVISOR AT THE END OF THE EXAMINATION.

Excellence
07

INSTRUCTIONS

Read all parts of the assessment task before you begin.

Select ONE of the following computer science areas:

- complexity and tractability
- computer vision
- big data
- computer graphics
- formal languages
- network-communication protocols.

Type your chosen computer science area in the space below:

Formal Languages

Begin your answers on page 3.

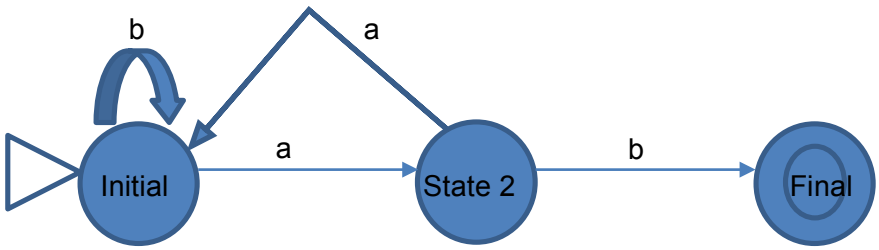
ASSESSMENT TASK

(a) Explain the key aspects of your chosen computer science area.

Formal Languages is a language type with strict set rules of how they are read. They are either correct or incorrect. Formal languages are used everywhere in computing and technology around us. Interpreters and compilers both use these in how they read their code. Because of their strict syntax, they can be understood by basic computer systems in order to do more complex tasks. A place where formal languages is used is in regular expressions, where you can have an exact command to try and match text or operations, creating an exact syntax.

(b) Explain the relevant algorithms or mechanisms that support your chosen computer science area.

A relevant algorithm of formal languages is regular expressions, often shortened to "regex". Regular expressions are a way of matching other text, in where they are set rules for the match. Finite state automata's (FSA's) can describe these expressions more simply. FSA's have an initial start state, paths between states and a final state. If an input finishes on the final state, the input is a match to the expression. Using JFLAP, you can model these with circles, arrows and labels. The initial state is shown by a triangle/arrow pointing to it, and a final state is shown by having 2 borders/circles. Say you had a FSA that started at the initial, and if you were to input 'a', it would move to the next state. From this state, if you were to input 'b', you would move to the final state. Say though, if you were at the initial state, and inputted 'b', you would loop and stay in the initial state. And from the second state, if 'a was inputted, it would loop back to the initial state again. It would look something like this:



So if something like 'ab' was inputted, it would pass and would match the FSA, as it moves from the initial state to the final state. If 'aabbab' was inputted, it would also pass, as it still finishes in the final state. However, if 'aabba'; was inputted, the input would not finish on the final state, so it would fail and not match. This shows the strict rules in formal languages, where it is either correct, or incorrect.

FSA's can also be translated into regular expressions. This FSA would look like this:

(b*(a{2})*)*ab

The '*' means 0 or more of that input.

The {2} means 2 (whatever is in the brackets many) of the input before it.

The () are to capture everything enclosed.

These can also be used in Ruby (Rubular) expressions, where the input (in this case, a's and b's) are in square brackets []. These can be used to find text in documents as anything can be the input.

(c) Explain how your chosen computer science area is used, implemented or occurs. Use examples to support your answer.

We can use regular expressions in Microsoft word to find words in mass text documents. The find window has a 'Wildcard' option that allows it to use the search like a regular expression. This way, you can find things like specific emails in a list of users of a site using a regular expression. Say you needed to find the email of someone you can't exactly remember the name of on a database. You can use regular expressions to search for emails that have "Bob" somehow in the email to find his email. Something like:

\w*[b][o][b]\w*[@]\w+[.](\w+|(\w+[.]\w+))

\w means any character of the English alphabet.

+ means to have one or more of that thing (whatever's on its left).

From this, you could find:

drycleaningbob@gmail.com

bobdaverson@office.co.nz

bobthomas@engineering.org.nz

"Ah, Bob Thomas was who I was looking for!"

From there, you can write your email to the engineering organization as you needed. This is one of the many ways that formal languages occur in our lives.

(d) Explain the key problems or issues related to your selected computer science area, AND how these have been, or may be, addressed.

Because of the nature of regular expressions being very strict, depending on the complexity of the expression, you can get errors in what matches. In ruby, you can have

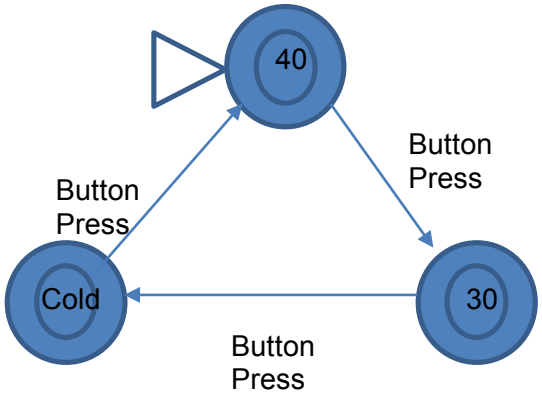
\d{1, 3}[.]\d{1, 3}[.]\d{1, 3}[.]\d{1, 3}

to match an IP address. Though this works for valid IPs like “10.251.9.23”, it will also match 999.999.999.999, which is not a valid IP in most cases. The way to fix this is to add to the complexity of the expression, but how far must you go to get the exact thing your looking for? You could have lots of ‘or’ statements (|) to the point where its 2 lines long, all to find exact IP addresses. And of course, as you make it more complex, the chances of errors occurring increases.

This problem can be addressed by only making the regex as complex as you need to in order find what you need in your current situation. If you are only looking for IP addresses starting with ‘10.251.’, then why not have the regex look for those exact numbers to start and add ‘\d{1, 3}[.]\d{1, 3}’ to the end of it? Its as complex as you need it in the current situation. Making it more complex can result in errors in what it does and doesn’t match.

(e) Provide a detailed explanation of how the technical capabilities and limitations of your chosen computer science area relate to humans. Use examples to support your answer.

Regular expressions are used everywhere, without us knowing. They are used in buttons for most machinery we use everyday, like washing machines, microwaves, vending machines and even our phones. In a washing machine, the temperature button can change its setting from 40 °, to 30 °, to cold, and back to 40 °. This is a regular expression, as you take the input of pressing the button, and it moves to the next state.



The regex for this is simply:

a*

where a is button pressed.

This can also be done for our phones. When we call phone numbers, the satellite must find that exact phone number, though some numbers will clearly be invalid and can be cancelled before making a search. A number like 249 3904 3950 3409 231 clearly is not a valid phone number (at least not in New Zealand it isn't), and can be rejected.

(f) Compare and contrast different perspectives on your chosen computer science area.

Say you were writing a python program, and you simply wanted to print “Welcome to Python” when it ran. The code for this is “print(‘Welcome to Python’)”. The format of this line must be exactly like this as to not get any errors. If the apostrophe was in the wrong spot the python interpreter wouldn’t know what to do with it, and would give a syntax error, though we can see what the line of code was trying to run. We can see that “print(Welcome’ to Python’)” has an error and that it was trying to get the terminal to print “Welcome to Python”, but the interpreter can’t, as it must follow its regular expression for how a print statement works. Though this seams the interpreter is at a disadvantage when it comes to reading its code, the strict rules around the language are very important to be able to write code in the first place. We would not be able to write code if the number of indents didn’t matter or if defining a variable only worked when the interpreter decided that it worked.

(g) What conclusions can you draw about your chosen computer science area?
In your answer, you could:

- explore less-obvious implications
- justify predictions that you make
- consider potential improvements
- suggest innovative and imaginative wider uses.

Formal languages are used everywhere in our society, hidden in plain sight. It can be found in any machine used today. Without such a language type, code would not be able to be written or run in the way it is now. What would magazine companies do if they had a typo throughout an entire magazine they need to print? Do they go through it manually? Checking every word to find the “Lewis” ‘s and change them to “Louis”? with regular expressions, you can check for every “Lewis” and change them accordingly with precision and speed. One of the many uses of regular expressions. They can be used in data bases in schools to find every year 9 student to send out a general email. Even Google as a search engine is really just regular expressions, looking for exact words in your search and finding them on websites. Life would be very different without regular expressions and formal languages as a whole.

Excellence Exemplar 2019

Subject	Digital Technologies	Standard	91908	Overall grade	07

Q	Grade	Annotation
		The candidate accurately and succinctly analysed Formal Languages within their report. They explained, with use of examples, the Algorithms and Mechanisms, and referred to content showing clear and comprehensive understanding. The candidate related their understanding to a problem, and clearly provided evidence of how this was addressed. They provided detailed explanations, which were backed up by examples. Some evidence of insightful conclusions was shown, however there was no clear critical analysis. Some of the candidate's examples showed only partial analysis. The candidate was able to generate diagrams in the exam that they had obviously prepared, practiced and rehearsed earlier for. This helped them illustrate their understanding very effectively. The diagrams were simple and / or effective. The candidate illustrated examples with regex and email addresses. Overall, the candidate's response to the assessment task warranted an Excellence grade.