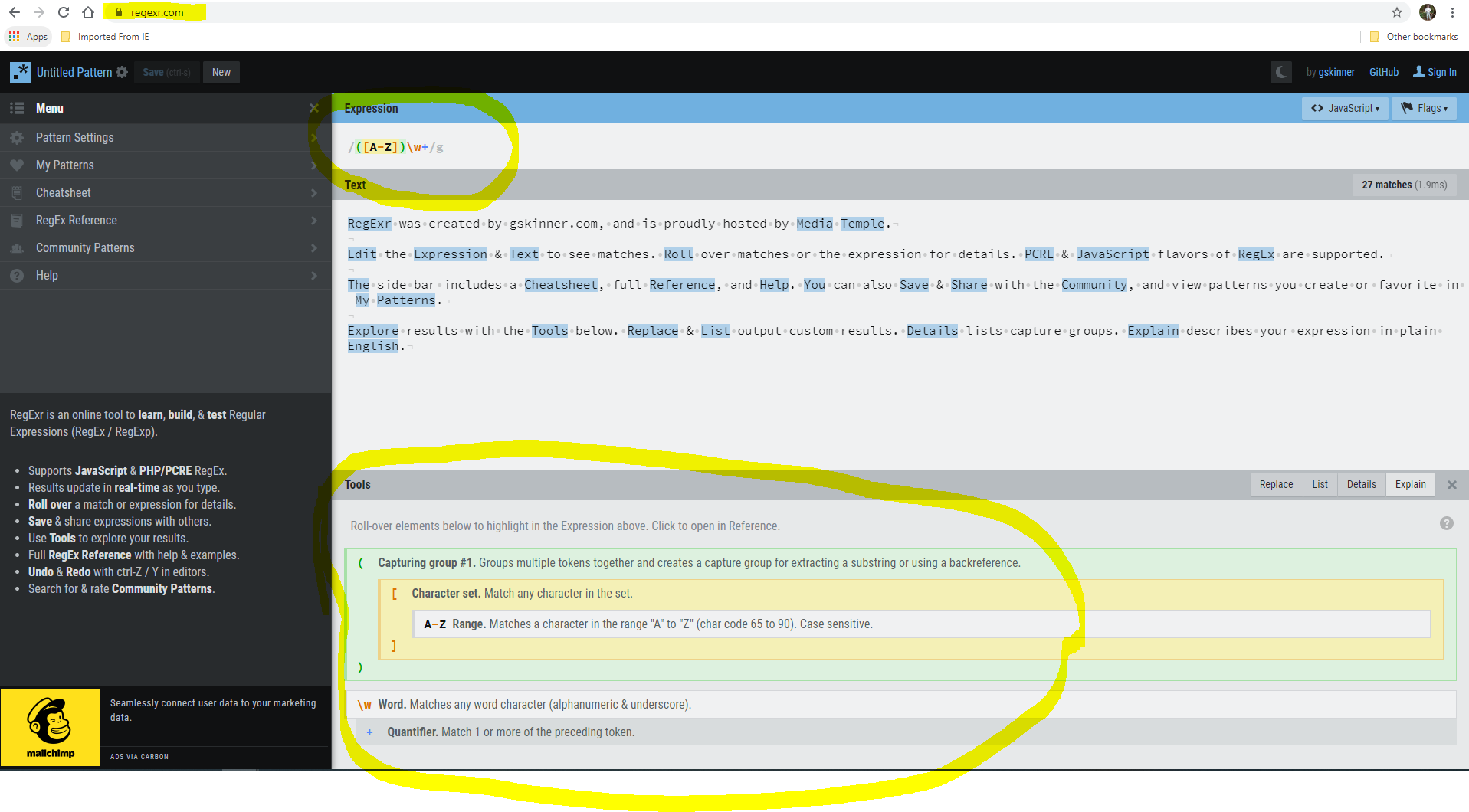
Has anyone heard of "regular expression / Pattern matching" before? (Please share your experiences. I love it.) “Regular Expression/Pattern Matching” is a critical feature in the Linux OS, and supported in all modern programming languages. In its essence, it is a rendition of Finite State Machine

One can get a jump start to it by learning it from this link:

<http://www.regexr.com/>

Example of the default session interface:



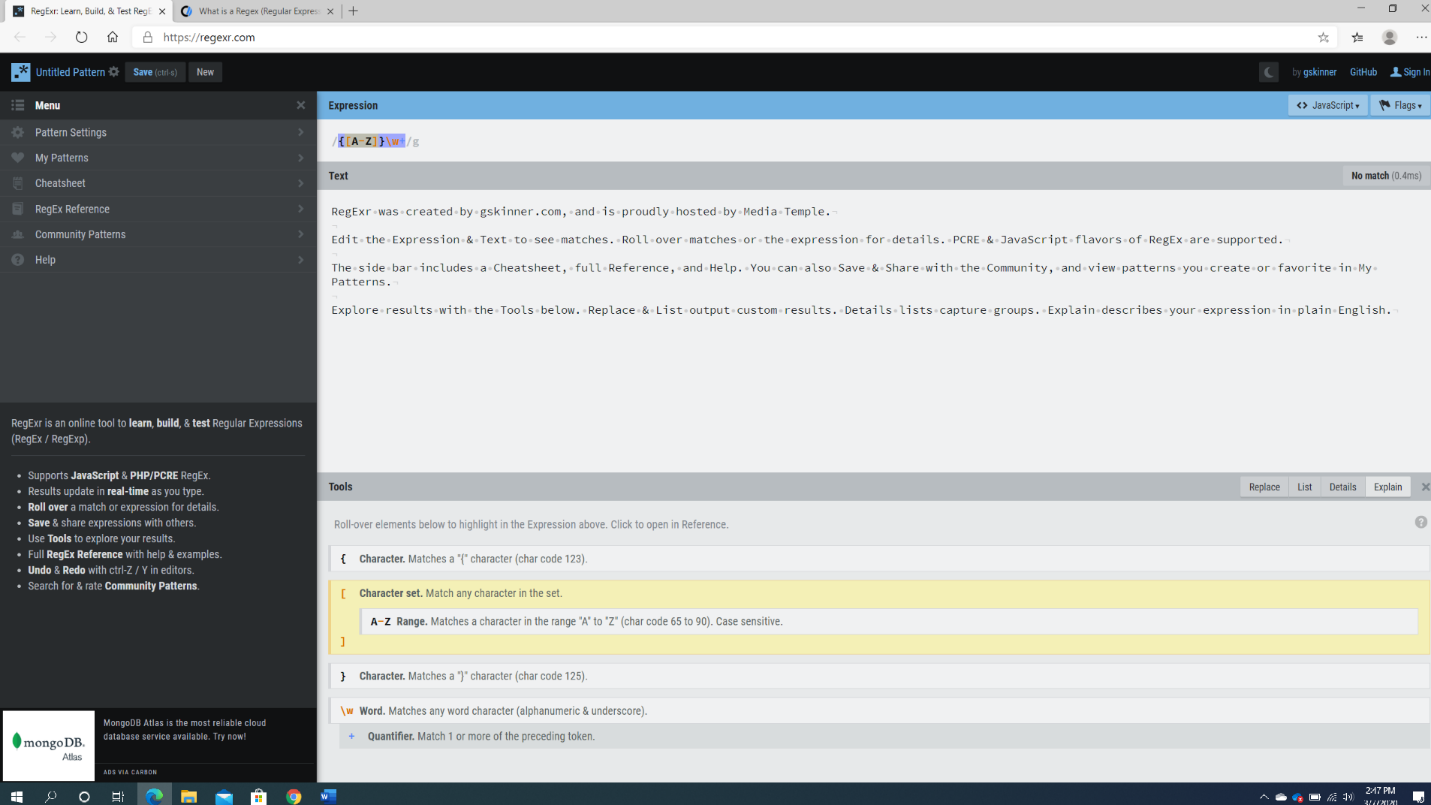
Lab Instructions:

1. What is your overall understanding of what the Website provides/does? (Please elaborate and capture your inputs below below)

After looking through the website and trying to understand different things, I was able to understand that the website “<https://regexr.com/>” is a online tool that will help in de-assembling the expression that is written into simple English text.

Example: -

The expression “{[A-Z]}\w+” can be divided in the below image as follows in the bottom of the image.



1. What does the expression “/([A-Z])\w+/g “ do?

The above expression “/([A-Z])\w+/g” can be divided into the following symbols.

* “/” indicates the start of a regular expression.
* “(” and “)” together groups multiple tokens and creates a capture for pulling out a substring.
* “[” and “]” match any characters in a given set. Meaning the data placed in above is range of “A” to “Z”.
* “A-Z” indicates the range of characters that match between “A” to “Z”.
* “\w” indicates the matches of any word character, either be alphanumeric or underscore.
* “+” is a quantifier matching one or more preceding tokens.
* “/g” indicates the end of a regular expression.
  + So the regular expression matches a range of characters among “A” to “Z”.

1. Per Website, let’s be sure its “Text” entry are as follows:

-----------------------------------------------------------

RegExr was created by gskinner.com, and is proudly hosted by Media Temple.

Edit the Expression & Text to see matches. Roll over matches or the expression for details. PCRE & JavaScript flavors of RegEx are supported.

The side bar includes a Cheatsheet, full Reference, and Help. You can also Save & Share with the Community, and view patterns you create or favorite in My Patterns.

Explore results with the Tools below. Replace & List output custom results. Details lists capture groups. Explain describes your expression in plain English.

------------------------------------------------------------

(Or as seen in RegExr.com default interface shown in the last image.)

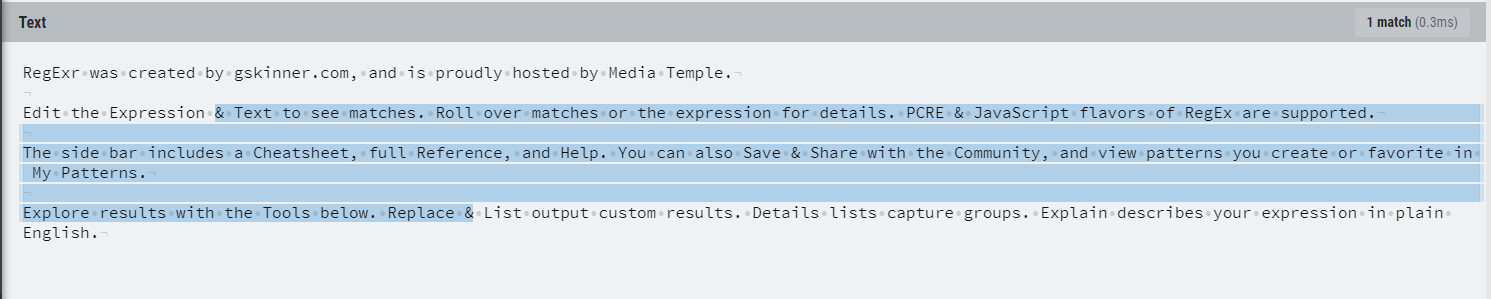
Question: What does the expression “/&.\*&/g “ do?

The regular expression “/&.\*&/g” indicates the following.

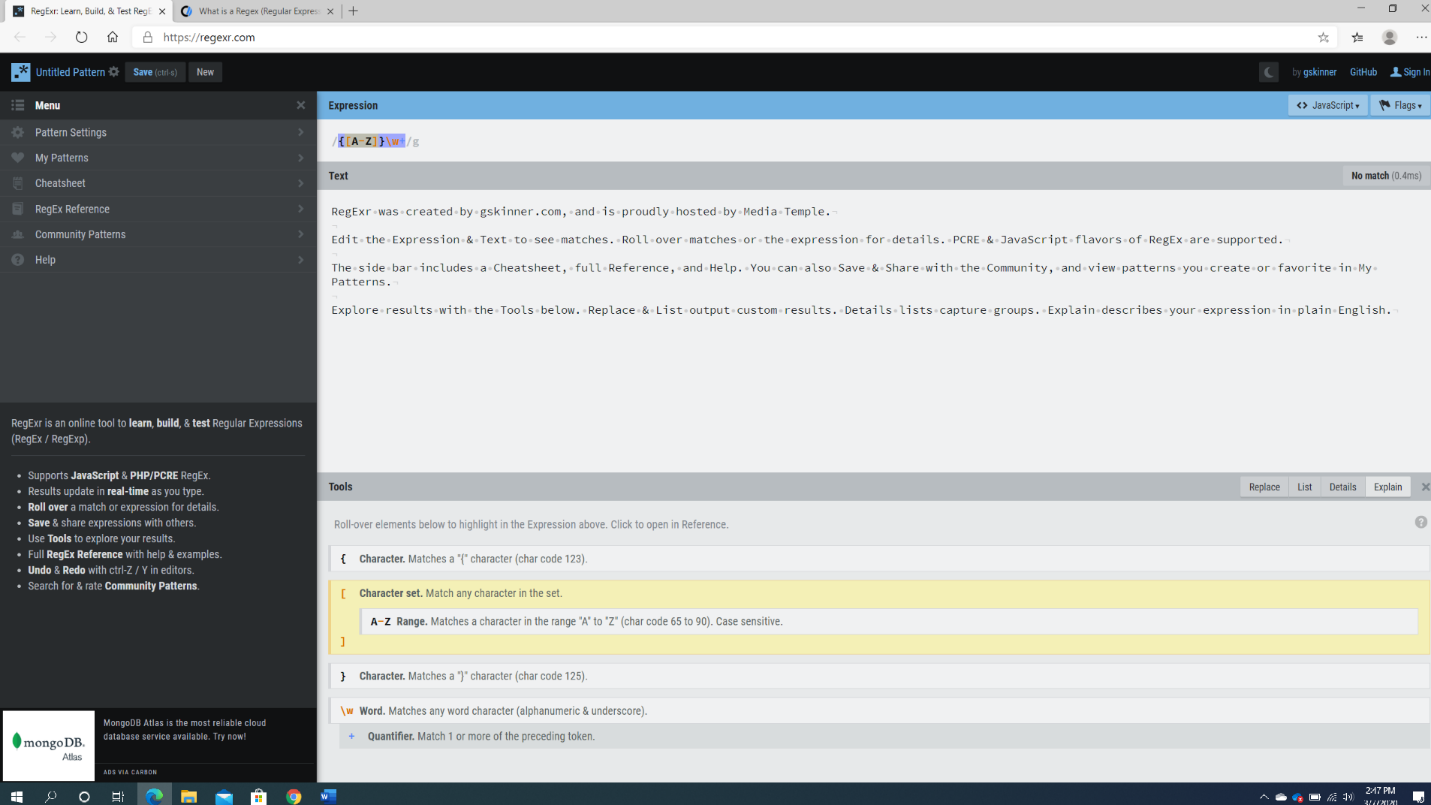
After the start string, the expression matches the “&” character with “.” Indicating that it continuous to match the characters except for breaking line.

The “\*” matches for 0 or more of the preceding tokens with “&” matching any number of characters till the end.

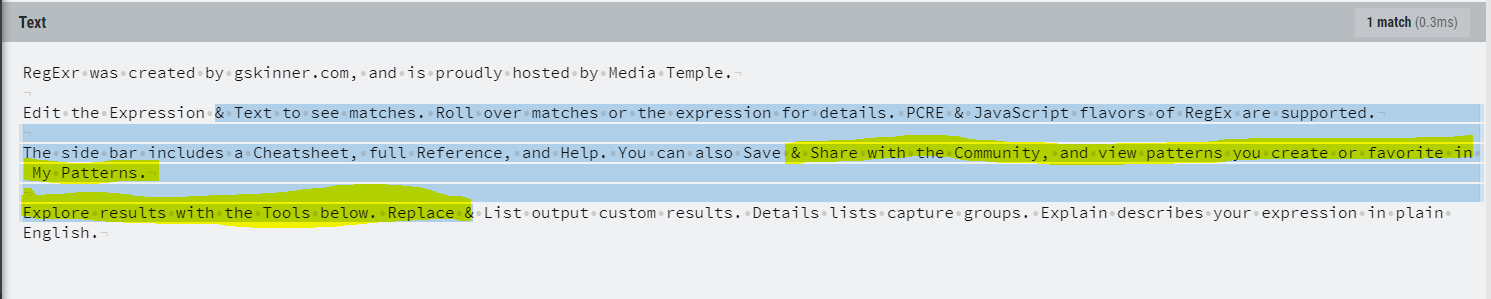
1. Now let’s accept “&” as a **tag**, and *the text strings between “&” can vary/dynamic*. Come up with the expression that matches the text string as shown below:

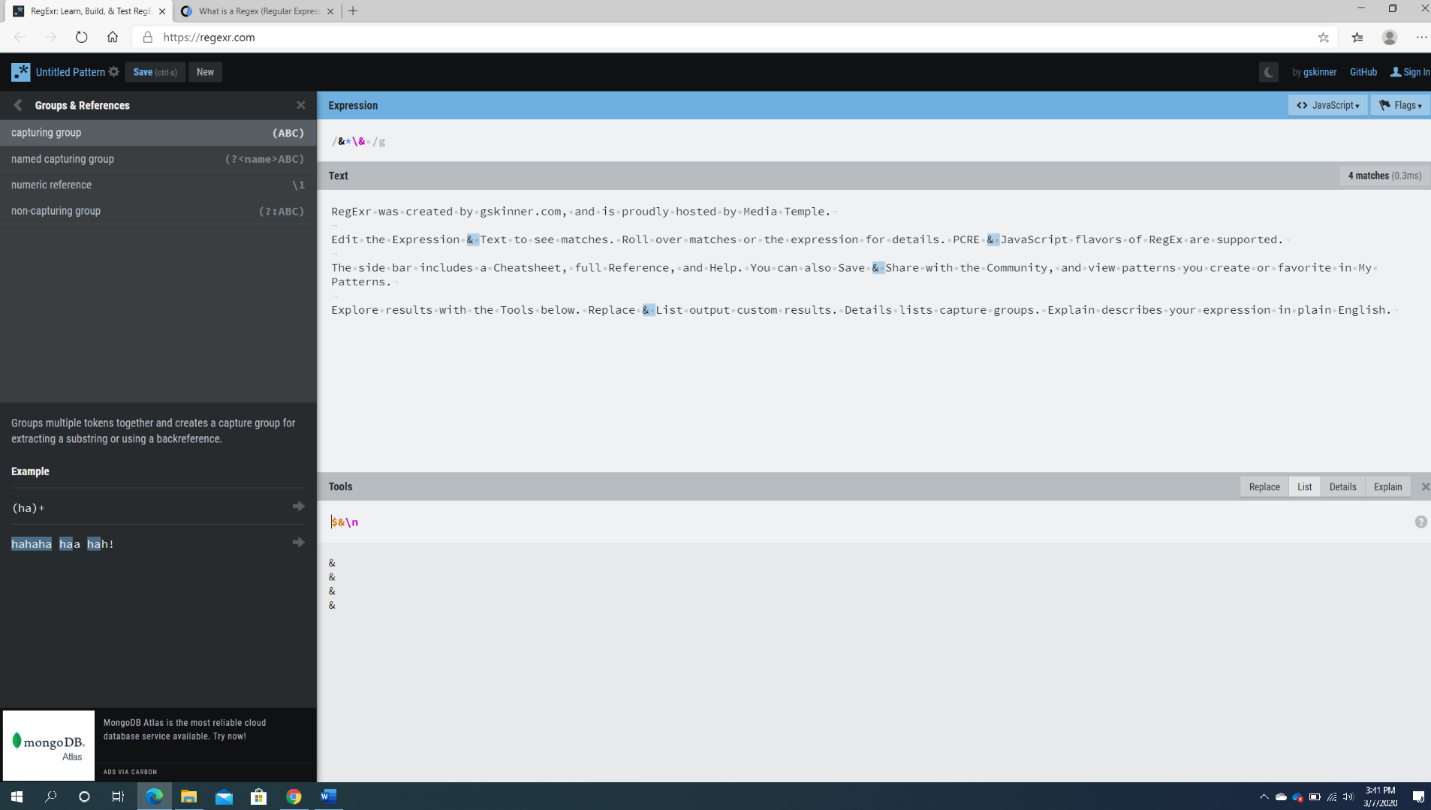


& Text to see matches. Roll over matches or the expression for details. PCRE &

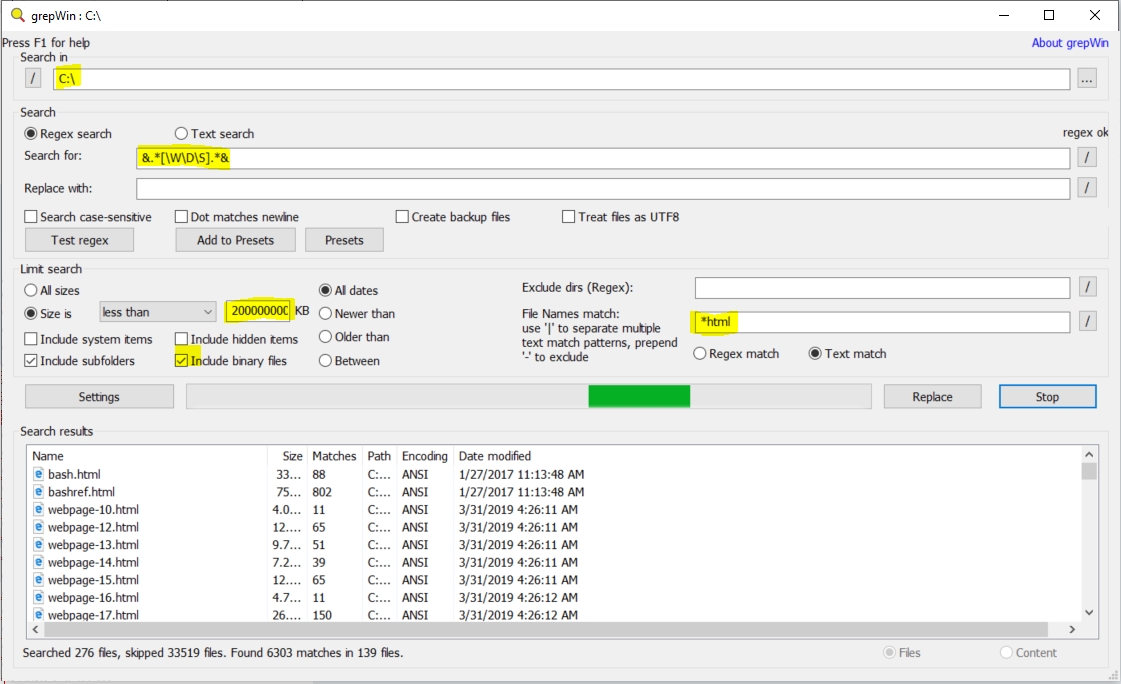


Bonus: Come up with the expression that matches the text string as highlighted below (*Remember, you cannot use dynamic/variable text string in your search pattern specification*):

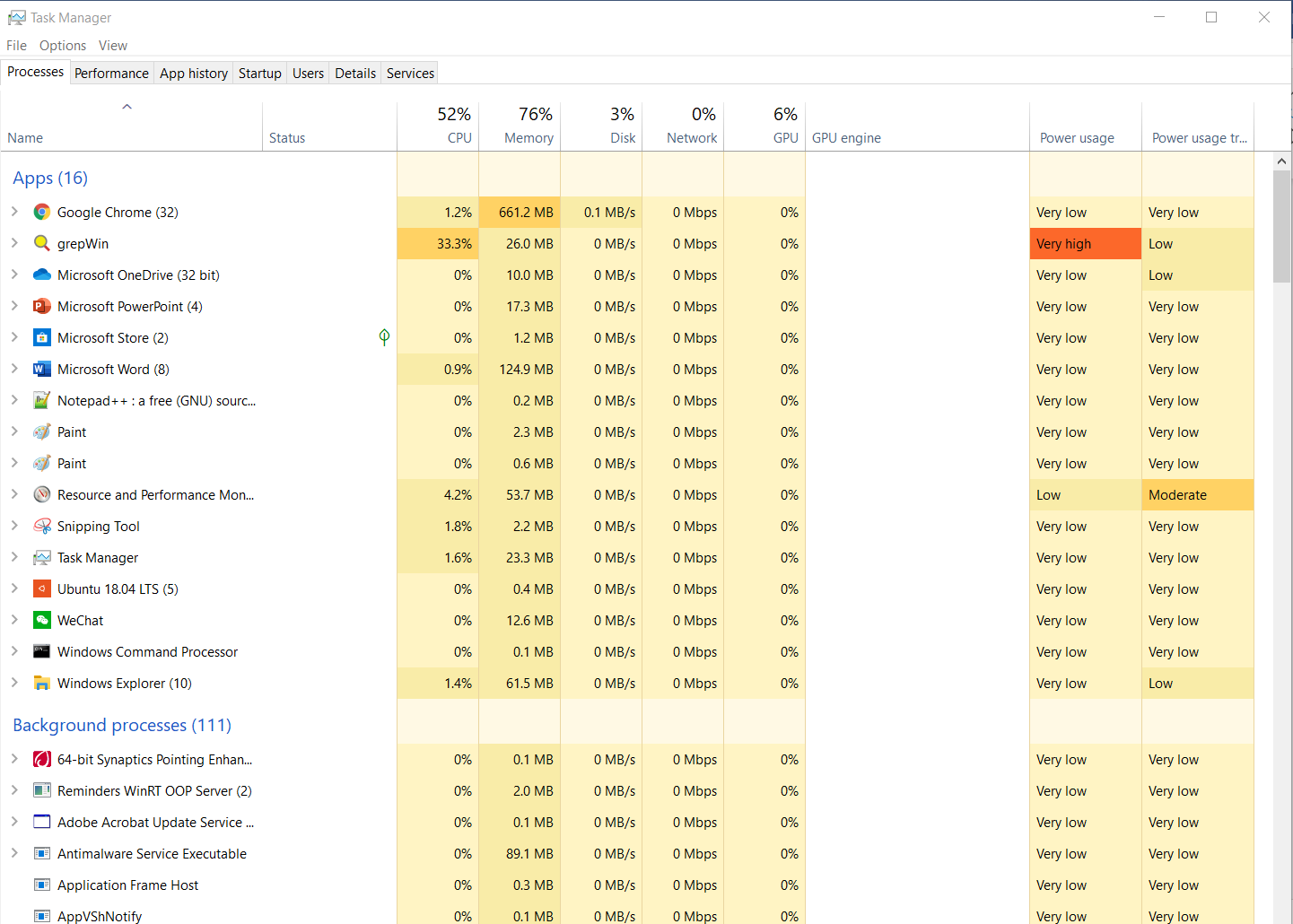




1. Launch “grepwin” and search for the text pattern “&.\*[\W\D\S].\*&” as shown below (note highlighted subtleties)



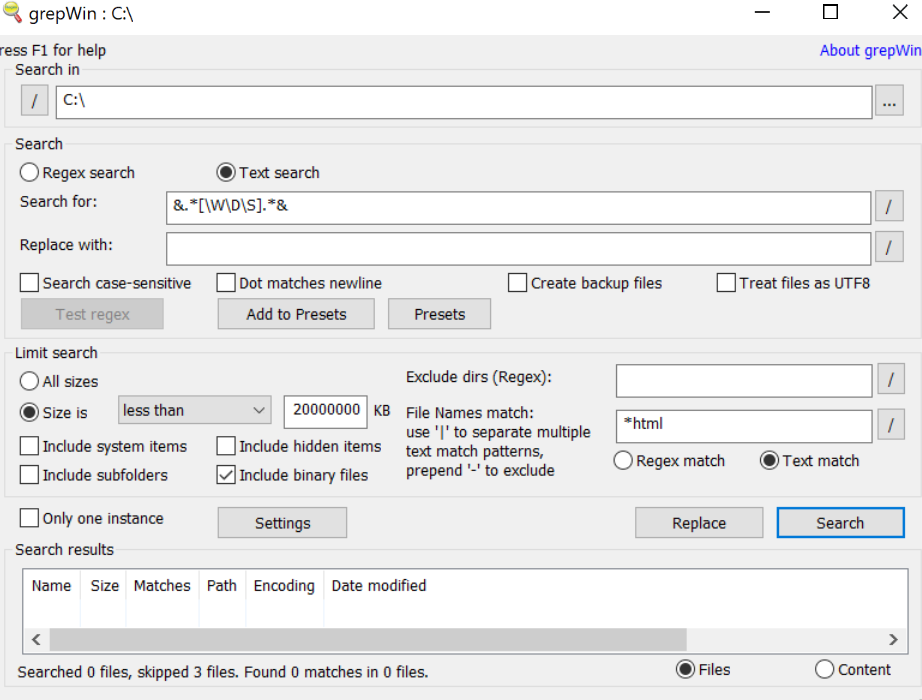
Monitor your CPU resource usage by “grepwin”. An example is shown below:

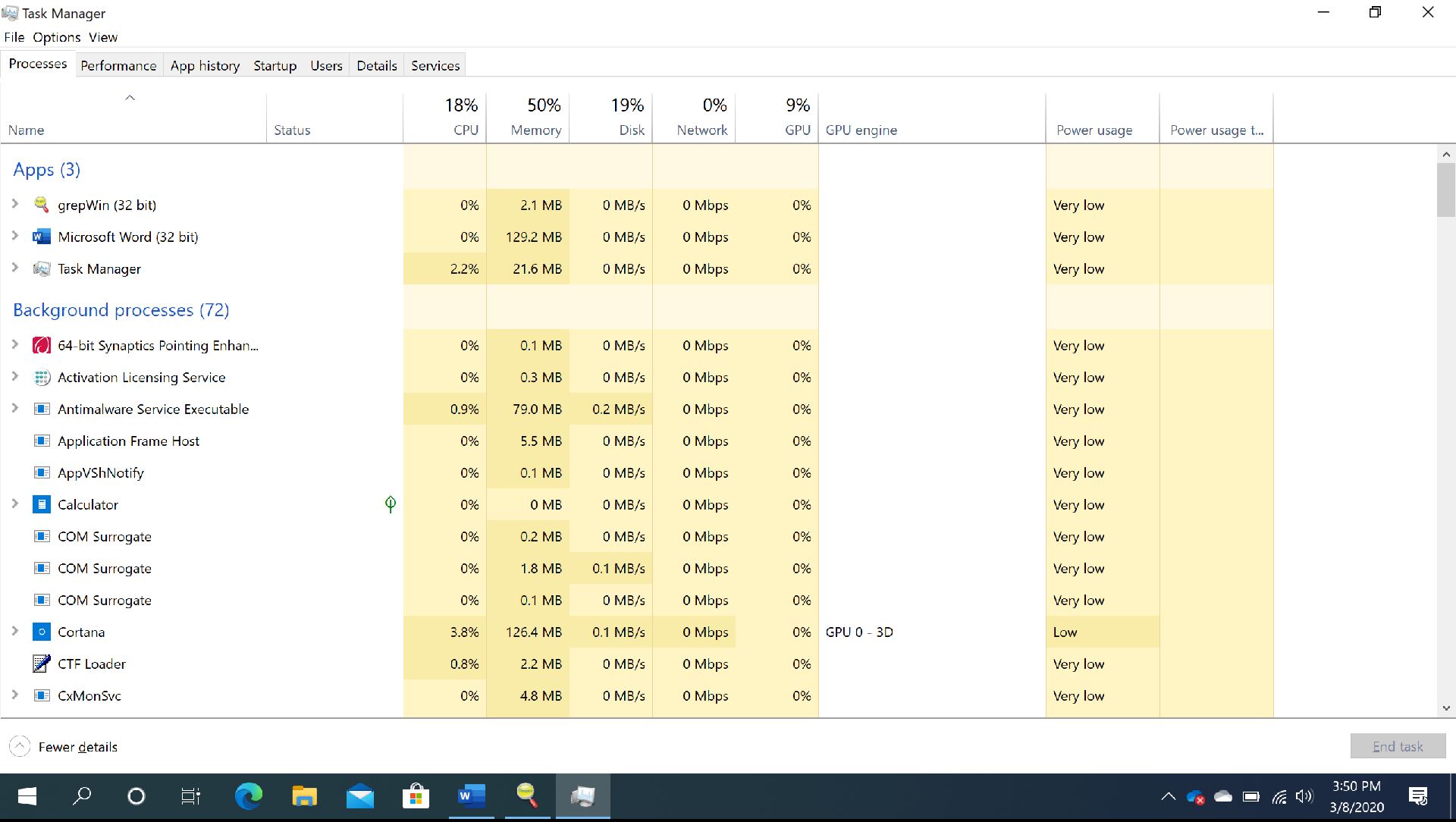


Questions:

1. Why this “simple” search operations take so long?

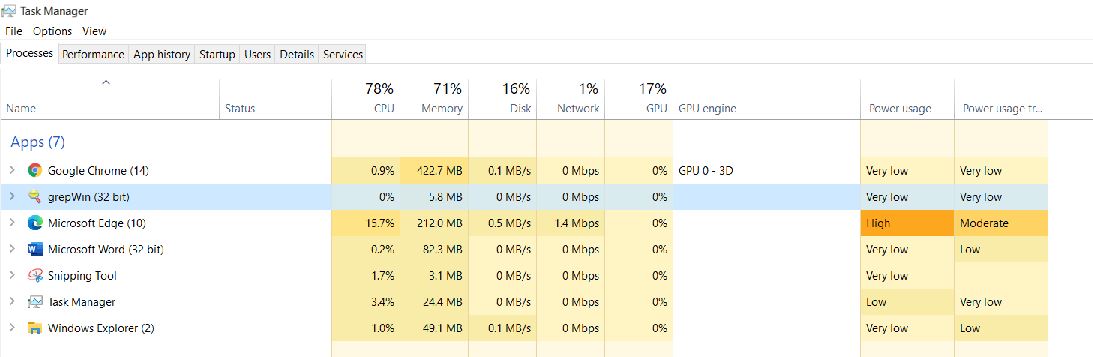
Well I have waited for more than 20 minutes and still could not find any search content. Following is the screenshot.





1. What computing resources are being heavily utilized by “grepwin”? (CPU, Memory, IO, Storage, Networks etc.)

Also since the search is being negligible, there is not much power usage. Following is a screenshot.



1. Try another “search pattern” of your utmost interest and make your observations.

Made the search for other search patterns but still the tool was not searching any of the patterns.

1. Any new ideas, breakthroughs to speed this up …?

I have deleted GrepWin and downloaded it again, did the basic reboot options but still I could not find any options of searching the content in Grepwin in the first place to increase the capability of increasing it search speed.

1. Your reflections:
2. Why Google does not support full RegExr search?

It most likely is certainly not an overwhelming procedure, however the primary explanation is presumably easier: an enormous level of Google's clients are most likely not extremely specialized or PC sharp, so they have no clue what regex even is. It doesn't bode well for Google to incorporate a regex include on the off chance that it won't be utilized a lot, since it's likely an exercise in futility and wouldn't be valuable to most clients. Likewise, I can't generally perceive how regex search would be valuable. It would be better if Google simply comprehended your question and gave all of you applicable outcomes dependent on exactly how well it got it. Additionally, it would be decently asset concentrated for Google to help ordinary articulations in search.

It might be helpful for the extremely, little portion of clients who realize how to utilize regexes.

To put it plainly, the arrival of speculation for actualizing regex scan simply isn't there for Google and that is the explanation they haven't upheld it.

To help a regex search, for a regex question, Google should coordinate against each character in each and every url that they list. Accepting a list size of 50 billion urls (Source) and a normal length of 1000 characters for each url that is 50 trillion calculations that they should accomplish for each and every regex search. That is enormous even by Google models. Also the specific foundation they would need to put resources into to make this productive.

The ebb and flow catchphrase-based hunt, interestingly, is served utilizing particular records and is requests of extent less asset serious than a regex search.

1. Why Windows OS does NOT support RegExr search? (Should it? E.g. When my search target is a very small sub-folder)

Similarly, for Windows OS, in the first place, it is beaten by Google search tool for faster computational power. Now that if they want to use Regex for searching, it would be a huge investment and the company had to suffer heavy losses when compared to use it in public market.

1. Do you find/have a better RegExr search tool besides “grepwin”?

There are multiple files besides Grepwin that we can use. Even if we wish to use them only without downloading them onto the local system. Some of them would be the following.

<https://regex101.com/>

<https://www.regextester.com/>

<https://www.regexpal.com/>

1. You are now asked to implement the RegExr pattern search in *digital hardware*. Let’s call this the Discrete Logic *“RegExr” Machine* (e.g. Verilog/FPGA Platform). Details:

The input to your design is a sequence of ASCII symbols.

Come up with the Finite State Diagram, implementing the RegEx specification: Search Pattern = “&.\*[\W\D\S].\*&”.

Questions:

1. How many states in your Finite State Machine (FSM) design?

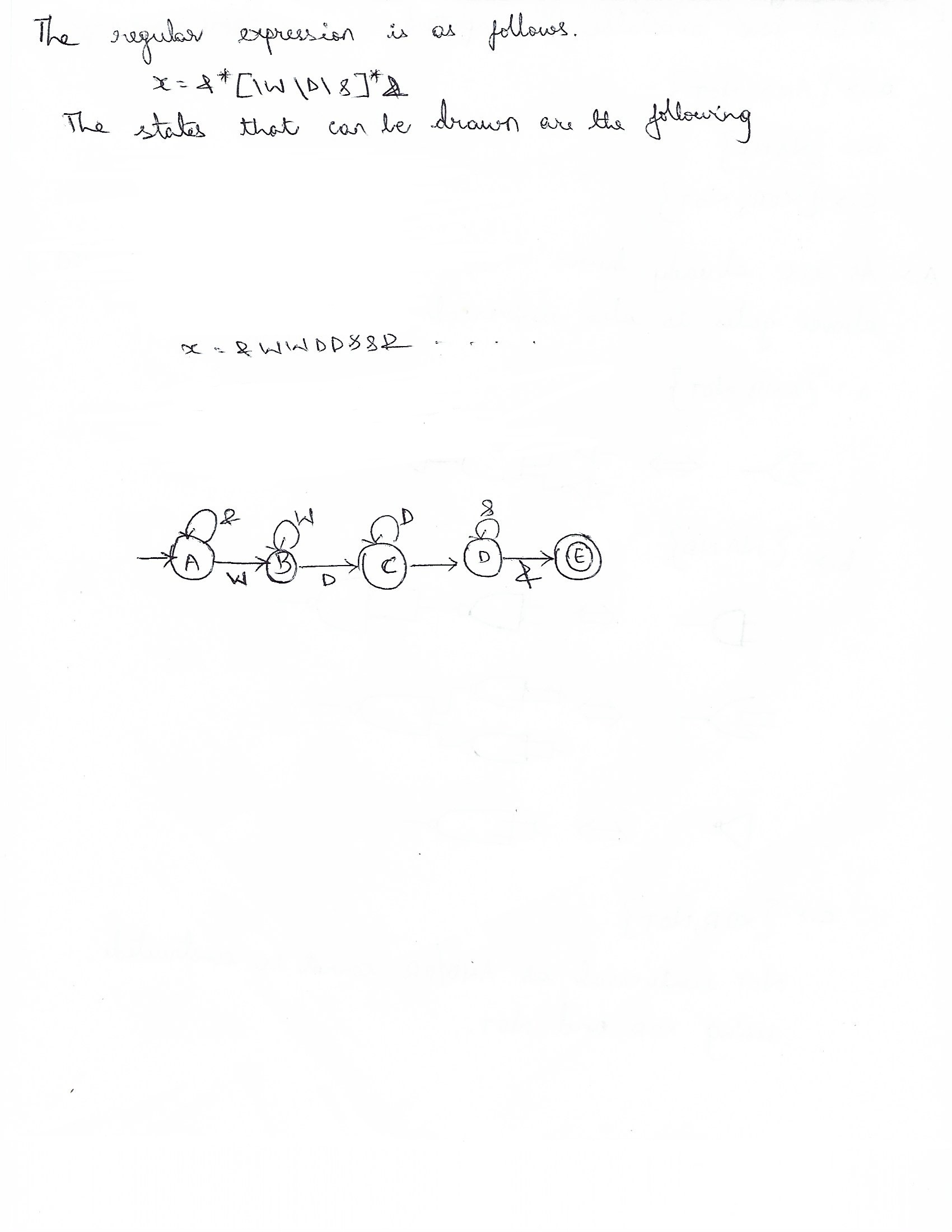
There are 5 Finite State Machine in according to mu design.

1. How many Flip Flops (Registers) to implement your design in discrete digital logic components?

In my finite state, I did not include any flipflops, however availed closure property.

1. Per <http://www.asciitable.com/>, Your ASCII symbol/input comes in 2 hex numbers (8-bits) format. How do you encode/decode the 8-bit binary inputs to your FSM design? (Use schematics/diagrams to capture your implementation approach.)

Please see the below picture for reference.



1. Oh well, you are being challenged to make a better design. We want the RegEx specification (Search Pattern) to be user defined. (In other words, let users specify it in a software program, or via a user interface.) We want to build a General Purpose *“RegExr” Machine!* Please explore and come up with your ideas/design to fulfill this requirement. (Hint: Arduino, Raspberry Pi Platform etc.?)

For the regular expression “/ \r\n[0-9]+\.[0-9]+\r\n\r\nA/”, the Arduino code can be as follows.

Serial.print(sTime);

//Serial.print(", ");

Serial.print("\t");

data = getdata(dataRead);

data = data>>4;

Serial.print(data);

//Serial.print(", ");

Serial.print("\t");

data = getdata(dataRead1);

data = data>>4;

Serial.print(data);

//Serial.print(", ");

Serial.print("\t");

data = getdata(dataRead2);

data = data>>4;

Serial.print(data);

//Serial.print(", ");

Serial.print("\t");

data = getdata(dataRead3);

data = data>>4;

Serial.println(data);

Where the regular expression is been sub-categorized for a better understanding.

/ \r\n[0-9]+\.[0-9]+\r\n\r\nA/

matches the character literally

\r matches a carriage return (ASCII 13)

\n matches a line-feed (newline) character (ASCII 10)

[0-9]+ match a single character present in the list below

Quantifier: + Between one and unlimited times, as many times as possible, giving back as needed [greedy]

0-9 a single character in the range between 0 and 9

\. matches the character . literally

[0-9]+ match a single character present in the list below

Quantifier: + Between one and unlimited times, as many times as possible, giving back as needed [greedy]

0-9 a single character in the range between 0 and 9

\r matches a carriage return (ASCII 13)

\n matches a line-feed (newline) character (ASCII 10)

\r matches a carriage return (ASCII 13)

\n matches a line-feed (newline) character (ASCII 10)

A matches the character A literally (case sensitive)

1. On the software programming model of the General Purpose *“RegExr” Machine:*

Specific Questions/Tasks:

1. Where/When to specify the RegExr “Search Pattern”?

As we can see in the above, a regular expression is needs to to specified at the start to design the logic accordingly.

1. How does your program execution begin?

Once the program is been designed, the compiler in Arduino will compile it and move forward to the next step, which is to run the program.

1. When does your program end?

Once the execution is been made, for external output, there are two ways for this. On method is that in the Sketch board, the Arduino will display the results or via connecting it to the external elements like lights or ultra-sensors and so forth, the results can be found out. (only depends on the type of device connected – if a light is connected, it will blow/pop the light as mentioned but a sensor is connected for measuring distance, it will be displaying the distance in the sketch.

1. What if the ASCII inputs come in intermittently?

Well for the regular expression I have chosen, there are no ASCII characters in place, however if there are any, there would be a slight change in the way the code is designed as new inputs would come into picture.

1. What is your hardware/software development platform?

(Hint: Explore interrupt driven programming/codes)

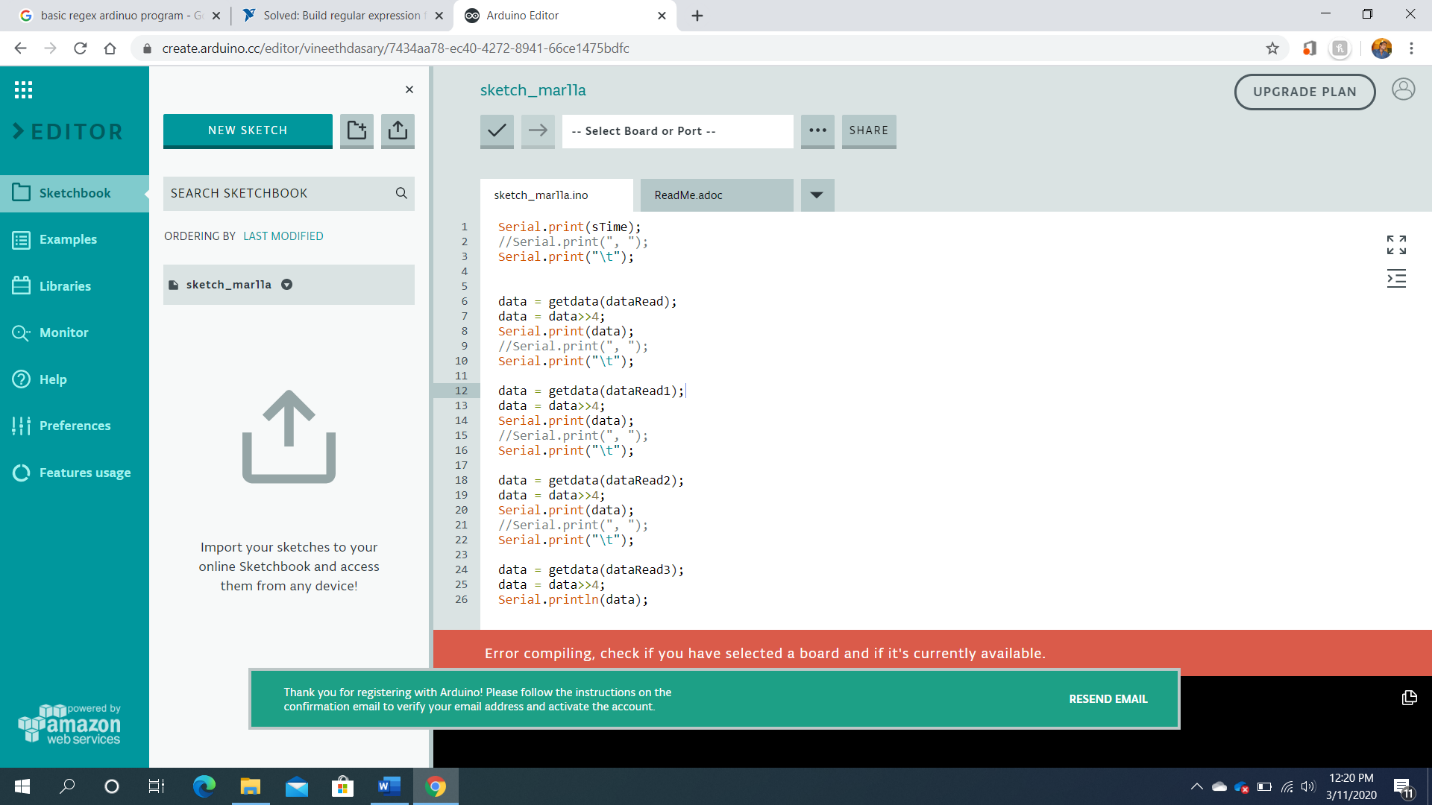
For the above code, the software requirements would be a Ardunio 1.8.12 downloaded.

For hardware requirements, laptop/computer with i5 processor, 4 GB RAM and some storage space would be needed.

1. Implement your source codes in the Arduino platform, and show that it compiles clean. Highlight your source codes where it fulfills the design goals.

For a program to run using Arduino, there should be a port connected to the computer or laptop for compiling and running the program.

As I do not have a port and a board with me to connect, the Arduino compiler compiler shows an error about that. Please find the below picture as reference.



1. Your preparations/ideas to debug/bring-up your General Purpose *“RegExr” Machine?*

Well I assume that before developing a Arduino program anytime, we have to first and foremost understand the requirements of designing it. Meaning to say is that we have to know the regular expression for which a design can be built, as for the Regex expression changes, also the design pattern should change to meet the requirements.

1. Questions:
2. Can you provide specific application scenarios (e.g. IoT, Internet of Things), where the *“RegExr” Machine* can be the solution?

Regular expressions are used in search engines, search and replace dialogs of word processors and text editors, in text processing utilities such as sed and AWK and in lexical analysis. Many programming languages provide regex capabilities either built-in or via libraries.

1. Elaborate the pros and cons on the two design approaches of implementing our *“RegExr” Machine*: General Purpose vs Discrete Logic.

Regex general Purpose: -

They can be laconic - a ton of complex data is compacted into a couple of characters

The mind-boggling conduct can be hard to structure and investigate. They are just helpful in removing 'normal data' - for example data that is given in a predetermined number of known organizations; and numerous genuine issues aren't 'normal dialects' - for example utilizing a customary articulation to attempt to parse HTML is a downright awful thought - HTML contains dreadfully numerous discretionary parts and exceptional guidelines. Customary articulations are very ground-breaking, yet are known for being "compose as it were." Once they go past a fundamental degree of multifaceted nature, good karma attempting to make sense of what they do.

Discrete Logic Regex: -

Discrete rationale configuration won't be completely eliminated. There will consistently be applications where utilizing a discrete rationale IC is ideal. As has been brought up, speed is a major preferred position, in spite of the fact that in a great deal of uses, the speed distinction is simply not unreasonably significant.

With regards to the plan stage, in the event that you structure a circuit that solitary needs to perform 2 or 3 basic rationale capacities at various focuses in the circuit, it will be smarter to utilize discrete doors, just to spare structure time on having to likewise compose the program as well.

For frameworks that need to accomplish increasingly complex rationale capacities, it is senseless to invest all the energy working out a fact table, at that point making sense of which rationale doors go where and so forth when you could simply compose a little program. Generally, the more sources of info mean the more entryways required and the more it takes to structure in discrete.

Where discrete rationale truly has the bit of leeway however is in learning. At the point when you are first finding out about rationale structure and how doors work and so forth, that is the place getting hands on with real rationale entryways and planning various capacities with discrete parts is incredible. Continuously a smart thought to get a comprehension of essentials. Along these lines, therefore, discrete rationale is continually going to have a spot right now. With respect to purchaser hardware? What's to come is certainly programmable.