Student and Computing Information

- Nicholas Phillips and Student ID: A031344011

- Programming Language Theory, Assignment 1.

1. Consider the language S*, where S = {ab, ba}. Write out all the words in S* that have seven or fewer letters. Can any word in the language contain the substrings **aaa** or **bbb**? Explain!

- S* = {λ, ab, ba, baba, bababa, abab, ababab, abba, baab, babaab, ababba, baabba, abbaab, baabab, abbaba}
- No, none of the words in the language can contain aaa or bbb substrings. This is because in S = {ab, ba}, ab and ba have succeeding different letters, hence not allowing aaa or bbb to appear as substrings in the language.

2. Let S = {ab, bb} and let T = {ab, bb, bbbb}. Show that S* = T*

- S* = {ab, bb} = ab+bb = abbb
- T* = {ab, bb} = ab + bb = abbb
  Or
- S* = {ab, bb, bb} = ab+bb + bb = abbbbb
- T* = {ab, bbbb} = ab + bb = abbbbb

- S* and T* are in fact equal to each other. This is because for T = {ab, bb, bbbb}, there is a set of 'bbbb' that can be generated from S = {ab, bb}. 'bb' would need to be doubled so that the sets can be equal to each other. There are examples above that show an explanation of how the two can be connected with each other.

3. Give a recursive definition for the set ODD = {1, 3, 5, 7, …}.

- R1: 1 is in ODD
- R2: If X is an ODD number and Y isn't an odd number, then if the expression x + y is greater than 0, then it will be equal to an ODD number in the set {1, 3, 5, 7, …}.
- R3: The only numbers in the element are those that can be produced from R1 and R2 from above.

4. Construct a regular expression to define each of the following languages over the alphabet ∑= {a, b}.

(a) All strings that end in a double letter such as aa or bb.

- (a+b)*(aa+bb)

(b) All strings that do not end in a double letter such as aa or bb.

- (a+b)*(ab+ba)

5. Describe (in English) the languages associated with the following regular expression.

(a) (a(a + aa)*)*

- The regular expression will create or make "a,aa,aaa,aaaa,aaaaa…" This means that each string created will begin with an 'a' and followed by another 'a' and will only create a string containing only 'a.'

(b) ((a + b)a)*

- The regular expression will create or make strings containing 'a,b' then following λ where 'a' will always be the even spot in the string.