**Assignment 5**

**100 Points**

1. **Show that the language L = {anbm: n and m are prime numbers} is not context-free (10 Points)**pumping lemma is true for context free languages, therefore if it is false, it is not a context-free language. First thing’s first, let’s define a string that we can pump.  
   n = 3, m = 5  
   aaabbbbb  
   We can break this down into three parts:  
   xyz where x = a, y = aa, and z = bbbbb  
   Now we ‘pump’ y  
   a aa aa aa aa bbbbb  
   the resulting string is n = 9, m = 5, where 9 is not a prime number. Because the pumping lemma is not true, this is not a context free language.
2. **Construct a Turing machine that accepts the language L = L(aaaa\*b\*) via Transition Digram, Transition function and Instantaneous Description for the string aaaaabb (15 Points)**A red marker on a white surface

   Description automatically generatedFigure 2-1: The above Turing machine must accept at least 3 a’s, followed by any number of a’s, and reaches it final state when it either receives a ‘blank’ indicating that there are no more strings to read, or where it reaches a state where it can only read b’s. In this turing machine I have replaced a’s and b’s with X and Y for the benefit of the instantaneous description to show what has been read.   
     
   TRANSITION FUNCTION:  
   (q0,a) = (q1,X,R)

(q1,a) = (q2,X,R)

(q2,a) = (q3,X,R)  
(q3,a) = (q3,X,R)

(q3,b) = (q4,Y,R)

(q3, BLANK SYMBOL) = (q4,Z,R)  
(q4,b) = (q4,Y,R)  
  
INSTANTANEOUS DESCRIPTION:  
(q0,aaaaabb) ->(q1,Xaaaabb) -> (q2,XXaaabb) ->(q3,XXXaabb) -> (q3,XXXXabb) ->(q3,XXXXXbb) ->(q4,XXXXXYb) ->(q4,XXXXXYY)

1. **What Language is accepted by Turing machine with the following transition diagram and give two strings which will be accepted by TM (5 Points)**

**Image**A diagram of a mathematical problem

Description automatically generatedFigure 3-1: The above figure accepts (bb\*a) + (ab\*). Two strings that might be accepted are:  
abbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbbb, or bbbbbbbbbbbbbbbbbbbbbbbbbba

1. **Design Turing machines that copies strings of 1’s. More precisely, find a machine that performs the following computation via Transition Diagram and Transition Functions (10 Points)**

**q0w |-\* qfww**

**for any w ϵ {1}+** The described computation literally says for a Turing machine with a given input state of one’s, make the final outcome the initial input string of one’s but again (yeah I had to write this down, notation is confusing!).  
I’m thinking the solution is for every one read, we mark that it is read, then we enter a state where we continue reading right until we reach a blank space, then we write a placeholder. Then we go back to the symbol furthest left indicating we have read it, and repeat the process until we encounter our rightside placeholder immediately after writing a leftside placeholder. Then we go right all the way replacing all placeholders, then we go left all the way replacing all left placeholders.   
A diagram of a graph

Description automatically generated  
  
Figure 4-1: Implementation of the above machine. States q0,q1,q2 handle the copying of the initial string, and marking what is read. States q3,q4 handle replacing the read and copy symbols with 1, q5 terminates the process when a blank symbol is read and represents the final state.   
  
(q0,1) ->(q1,X,R)

(q0,Y)->(q3,1,R)  
(q0,BLANK) ->(q5,BLANK,R)

(q1,1) ->(q1,1,R)  
(q1,Y) -> (q1,Y,R)  
(q1,BLANK) ->(q2,Y,L)  
(q2,1) ->(q2,1,L)  
(q2,X) -> (q0,X,R)  
(q3,Y) ->(q3,1,R)  
(q3, BLANK) ->(q4, BLANK, L)  
(q4,1) -> (q4,1,L)  
(q4,X,) -> (q4,1,L)  
(q4, BLANK) ->(q5, BLANK, R)

1. **What language does the following unrestricted grammar derive (10 Points)**

**S -> S1B**

**S1 -> aS1b**

**bB -> bbbB**

**aS1b -> aa,**

**B -> ג**

OK, let’s look through the following language. It starts with S, which gives us S1 and B. B can become nothing. Neat. S1 can ONLY put an A and B on either side of itself, while replicating itself. Once it has put an a and b on either side of itself, it can consume one of the bs and simply become aa. Therefore we know that there can exist a language with 0 bs and only 2as. A single B can combine with the combination of a’s to create 3 bs, and a B. This means the B’s can endlessly multiply without further input, so we know that the language can have 0 or more bs, so we can write down B\* with some certainty. There will always be at least 2 a’s in order to get rid of S1, and for every a beyond that there will ALWAYS be an extraneous b.  
  
aa(a^n b^n) b\* where n>=0.

1. **Describe two variations of TM machines and give an example of their transition functions (10 Points)**The multi-tape turing machine is a turning machine with two read-write tapes. The function has both current reads of the tape as an input, and both writes of the tape and both tape directions as an output. The transition function inputs both at once so it looks like:  
   (q0, a,b) -> (q1, c,d, L, R)  
     
   The stay option turing machine is just like a regular turning machine, except there is a movement option of stay instead of move L or R. The transition function is identical to the regular turing machine, except the STAY command is used.  
   (q0,a) -> (q1,b, STAY)
2. **Describe the differences between recursive and recursively enumerable languages (10 Points)**Recursive languages are languages that can be accepted by a turing machine, but halt and accept on every string within the language, it will halt and reject any string not within the language. By comparison, recursively enumerable languages may run forever if a string is not within the language, but will halt if the string is within the language.
3. **Given the following context-sensitive grammar for the language L={anbncn: n>=1}. Show the derivation of a4b4c4 (10 Points)**

S → abc|aAbc,

Ab → bA,

Ac → Bbcc,

bB → Bb,

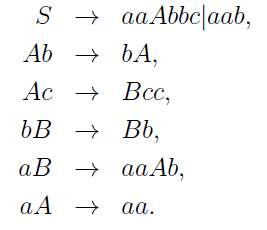
aB → aa|aaA

**S  
aAbc**

**abAc  
abBbcc  
aBbbcc  
aaAbbcc  
aabAbcc  
aabbAcc  
aabbBbccc  
aabBbbccc  
aaBbbbccc  
aaaAbbbccc  
aaabAbbccc  
aaabbAbccc  
aaabbbAccc  
aaabbbBbcccc  
aaabbBbbcccc  
aaaBbbbbcccc  
aaaabbbbcccc**

Look I am too tired to write out the step used for each one, but the theme is that the write head moves to the right until it reaches C, which adds a new b and c, then it cycles over to the left side and creates a new a, before repeating the previous step until you decide to stop.

1. **Given the following context-sensitive grammar, what language can be recognized? (10 Points)**

Seeing context-sensitive grammar makes me appreciate Chomsky normal form. OK, let’s read through this. S creates the initial thingy, you can stop there with at least two a’s and one b. Neat. Otherwise you end up getting aaAbbc. You could stop there, and turn aA into aa. That would give us aaabbc. Otherwise the read head that is A moves right until it encounters C, which then generates an additional c. The read head then moves left and spits out an additional a and b. This then returns to the option to eliminate a by placing down an additional a. This seems to mean that (abc)\* and there’s at least two a’s always.  
  
aa(abc)\* is the language that is recognized.

**10. Find context-sensitive grammars for the following language (10 points)**

L = {anbna2n : n >= 1}

If we read the above grammar, it reads that there must be one a for every b, followed by two bs. This must be true at least once.

S -> aAbaa | abaa

Ab -> bA (move the read write head to the right)

Aa-> Bbaaa (generate two new a’s, one new b)  
bB -> Bb (move the read write head to the left)  
aB -> aaA | aa (generate a least one new a, or terminate cycle)