Saim Ali  
  
1.   
(a) Min operation  
Text, letter

Description automatically generated  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
  
(b) Quotient operation  
Text, letter

Description automatically generated  
  
2.   
(a) Show that PRIMES is not regular  
Text, letter

Description automatically generated  
  
(b) Show the language COMPOSITES is not regular  
zzzzzzzzzzzzzzzzzzzzzzzzzzzz

3.   
(a) Prove/Disprove:  
There exist two languages that are both not regular, but their union is regular.  
  
**Prove-**   
Text, letter

Description automatically generated  
  
  
(b) Prove/Disprove:  
If L is any subset of a regular language R, then L must be regular.  
  
**Disprove-**  
Text, letter

Description automatically generated  
  
  
4.  
(a)  
For this question we must design a Turing Machine for the language which accepts a^i b^j c^k only when I =/ k or j = k. In order to do this with a Turing Machine I created the state diagram in JFLAP. The state diagram is shown below. To start my state diagram I first constructed a Turing Machine which accepted   
a^i b^j c^k only when i =/ k. To do this I changed all input “a”s to x, “b”s to y, and “c”s to z. For I =/ k I started with changing the first a in the input to x then iterated right until I saw a c which I changed to z, the diagram below does this until all a’s have been changed to x. Once this action has been completed, I will shift all the way left until a blank space appears and then I go right until I see a “c”; if no c appears that means all c’s have been changed to z for every a for x which leads to a reject outcome. If a c appears we will accept since I =/ k has been fulfilled. After completing this portion of the TM I branched off from q2 on my diagram below to accept any input j = k. To achieve this I changed all y’s back to b and all z’s back to c. Once this has been completed the machine will then find the left first empty space and then iterate right to search for any available z’s or y’s; if any are found we will reject since the condition has been contradicted. If no z’s or y’s are found that means for every b there is a c in the input meaning we will accept the input. For my TM I also considered inputs where a, b, or c could appear 0 times in the input. This resulted in me using a total of 10 final states to account for any input given to our TM with the absence of either a, b, or c. The final state diagram for this problem I constructed with 5 inputs is shown below.  
  
Test strings (3 accepts, 3 rejects):  
Letter

Description automatically generated with medium confidence  
   
State diagram:  
 Diagram

Description automatically generated  
  
(b)  
For this question we must construct a Turing Machine which has k number of c’s where k = I + j; a^I b^j c^k. To solve this problem I started my machine with traversing the input and changing one c to z for every a to x we encounter. After doing this, I did the same thing as I changed one c to z for every one b to y. After completing these two crucial steps I look for any a, b, or c’s present in the input and if there is any I will reject the input. On the other hand, if there is no remaining a, b, or c’s this will indicate all of the individual letters have been added correctly to form k number of c’s; the machine will accept the input. I also added factors to account for inputs where there is no a, b, or c’s present. I have shown the test strings as well as the state diagram below.

Test strings (3 accepts, 3 rejects):  
A picture containing table

Description automatically generated  
  
State diagram:   
Diagram

Description automatically generated  
  
Machine Schema:  
The machine schema asked for by this homework question was not fully covered in class in time for the due date, but I will attempt to provide one with the information I found through my readings (Lewis & Papa). I believe a machine schema can be interpreted as a makeup of different touring machines coming together to form one to complete a function. So, in this case, M1 will serve as the TM which will change one c for every a in the input string. M2 can be serve as the TM which will be changing one c for every b in the string. Lastly, M3 will be used as the TM which will traverse the tape to find any a, b, or c’s which were left unadded therefore resulting in either an accepting or rejection state. These 3 TMs will form the machine schema of this problem which has k number of c’s.  
  
5.   
  
6.   
Explaining why universal finite automata cannot exist:  
When considering a universal finite automata X which is assigned a language which needs more states than the number of states in X, we must expand it since its unacceptable. This leads to a contradiction: we need infinite states to achieve this but a **finite** automata cannot have infinite states.