Jack Stevenson

Activity 6-1: This Turing Machine is built off of an existing Turing Machine for multiplication. The machine replicates the input with a separating symbol and then multiplies the two values together to produce the square. Q0-Q9 handle the duplication of the value and the rest multiply the values together. Q1-Q9 read every 1 in the input, add a $ on the end, copy each 1 while keeping track of the original and then cleaning the tape.

1a: This machine marks the beginning of the string with a $ and marks sets of a's and b's until reaching the end. When the first a is encountered, it is changed to an x and the head continues down the tape until a b is found. The b is replaced with a y. The head then goes back to the beginning and continues counting pairs.

1b: This machine finds the middle of a string and then checks each character in each half of the string. It counts from each end, replacing a's and b's with x's and y's. Once a single a or b remains, it is replaced with a j or a k. This serves as the head of the second string. The first character in the first string is matched with the first in the second, and if the match, the head of the second string is moved back. The first character is deleted. At the end, if the head of the second string is adjacent to an empty space, the string is accepted.

2a: This machine removes a 1 on the right side of the - sign and then removes one on the right until it runs out of 1's on the left. Then it deletes the - sign and produces the result.

2b: This machine checks if there are five or more 1s in the string. If so, it removes five and returns the result. Then it checks if there are five of more and repeats until there are four or fewer 1s in the string. This is the end result.

3: We know that recursive languages are closed under complement and union. This is because the complement of a recursive language's Turing machine will accept the strings the original language did not accept and will not accept the strings that it did. The union of two recursive languages can be thought of as the combination of two Turing machines. We define the nor of two languages as the union of their complements. Both complements will remain recursive and their union will as well. This shows that recursive languages are closed under the nor operation.

4: Let us require that a Turing machine may only halt in a final state. This means that the Turing machine may only process input within the recursive language that it represents. For instance, if a string outside of the language is used as input for a standard Turing machine, the machine would be expected to halt in a non-final state, indicating that the string is not within the language. However, if a Turing machine may not halt in a final state, there must be a deterministic transition for every value between states (i.e. in a language consisting of a's,

with input consisting of a's and b's, we would need to handle b at every node even though it will never be accepted). This not only adds unecessary complexity but also allows for an infinite loop to occur. In fact, it would be expected that every invalid input will result in an infinite loop. This makes the modified Turing machine incredibly impractical for determining whether a string exists within a language.