**Notes:**

* ~~Need to implement tape as an ArrayList instead of just an Array.~~
* ~~Deal with cases where the turing machine goes off the infinite end of the tape by using \_.~~
* ~~Need to create a print method for the output.~~
* ~~Need to test the Turing machine against StacsCheck~~
* ~~Need to implement a better implementation of creating a turing machine with a description and a starting type rather than leaving everything inside the constructor.~~
* ~~Now just need to implement task 2 binadd.~~
* Carry over works by replacing the index on the tape with t to signify that the value is a temporary one. When the machine sees t, it knows to use it as a 1 and replace it with an x after reading it. This value is then added to the next readable value of number 2. For instance, with 101#100#011 is read out after checking the first index as x01#t00#x11.
* Binary addition implementation is wrong since it does not check if the rest of the number is complete.

**ParenTM**

* ParenTM works properly by replacing the initial character with the letter “s” to signify the start of the tape, then runs along the tape until it reaches the other end where it starts scanning for any opening brackets from right to left. Once it hits an open bracket, it changes state and goes right until it hits a closing bracket, signifying that there is at least one balanced bracket in the tape. The TM then repeats the process until it clears the tape of all brackets which determines whether the tape is balanced or not. At any point during the scanning process, if the tape hits the blank character at the right end of the tape, it is determined as unbalanced and therefore rejected.

**BinAddTM**

* In contrast to ParenTM, the design of the Binary Addition Turing Machine is slightly more complicated as it requires quite a bit more states and quite a bite more rules related to them. However, it is designed in a similar manner where the initial character is marked as a special character that allows the machine to find its way back to it. To begin the check, the value of the initial number is taken and added to the value of the second number at the same index. For instance, if the first number is 101 and the second number 011, then the values to be added will be 1 and 0 as they are the values at the first index of their corresponding number. Once the sum of both values has been attained, the value at the same index of the final number will be compared to the sum. If the value is valid, the number is crossed out and the tm will go back to the beginning of the tape to repeat the same checking process until the values of all three numbers are crossed out.
* This implementation works perfectly fine until there are is an addition sequence wherein a consecutive number of 1’s needs to be carried over to the value of the next index. To deal with this, a new character “t”, and state “carryOne”, has been implemented to signify and look out for a carry one. When a 1 is added to another 1 in the second number, instead of crossing the 1 out, a t will be placed in its stead. This will be used to signify that the value is a temporary value and will be added in during the next cycle. After setting the temporary value, the Turing Machine can continue normally. Once the TM returns back to the t in the next state, it will be treated as another 1 and be added to the total sum of the current index.

**PalinTM**

* One of the custom Turing Machines that was implemented, is a TM that can check if a string consisting of X and Y characters is a proper palindrome. Its implementation uses a similar method to the Balanced Parenthesis Turing Machine and goes across the tape crossing out matching characters at the beginning and end of the tape. This is done by checking if the first non-empty character (empty characters are signified by a -), at the start of the tape has a matching counterpart at the end of the tape. This is done repeatedly until the tape is accepted, wherein there are no more characters on the tape aside from the start symbol s (**Figure X**). If the first non-empty character at the beginning of the tape does not have a matching counterpart at the end of the tape, the tape is rejected.

**MakePalinTM**

* The second custom Turing Machine is one that creates a palindrome from a string of X and Y characters. This allows the previous PalinTM to check the validity of the palindrome and vice versa. To successfully implement this, the TM pastes a mirrored copy of the current tape at its end to create a palindrome. It does this by reading the initial character and setting it to either 1 or 0 to represent X and Y respectively, but it is also used to signify the position of the starting character. After reading and marking the starting character, the machine will go to the end of the tape and copy and paste the last character to the end of the tape. A character that has been copied, aside from the initial character, is marked as x or y. Once appended to the end of the tape, the TM will return to the end of the original tape and look for the first viable character **(Figure X)**. This process is repeated until the machine copies the starting character to the end of the tape. After appending the final character, every marked character (e.g. x, y, 1, and 0) will be reverted to their original state, thus, leaving a palindrome sequence.

**Tests:**

* Use screenshots of the tape to show the effect of the TM on the tape.
* Also show the number of moves in the screenshot
* Make use of graphs to show the difference and the increase of the number of moves as n number of symbols increases.
* Show complexity examples
* Show the output of the tape







 