About the algorithm,

* The data in spreadsheet has been converted into JSON and stored in the variable *data*.
* ***GetSortOrder(props)*** – This function has been written to sort the arrays with dictionaries by passing the dictionary item as *props* according to which the data needs to be sorted.
* The data is then sorted and stored in *sortedData* according to *writerid* so that we get all the tales written by a writer together.
* ***keyValuePairWriters()*** – This function takes the data stored in sortedData and takes out all the distinct writers with the tales written in this manner {writerid: somewritterid, tales: [array of tales written]}. This is then stored in *listofWriters.*
* ***sortLength()*** – This function sorts listofWriters in descending order according to the number of tales written by the writer. i.e. the writer with the most number of tales would be on the top and the writer with the least number of tales would be the last element. Optimized Bubble sort algorithm has been used here. We can also use Quick sort instead to make it more optimized by reducing the time complexity.
* ***makeSchedule()*** – This is the most important function as this makes the schedule for publishing the tales.
  + As the all the tales are of equal quality, we would iterate on the writerid. i.e. each writer would go public at least once given the number of writers is lesser than the number of days. That means one tale of each writer would be published first then second tale and so on. Coming to the approach, *talesPublished* is where the schedule is stored.
  + First, we check if the writer is already present in the array, if not then it means that none of his tales has been published yet. So, we push the first tale of the writer. If the writer already exists in the array, that would mean at least one tale of this writer has been published. We increment the writer, then we check if the tale is present or not. If it is then we increment the tale.
  + Initially we have also pushed an undefined tale so that undefined tales doesn’t get pushed. Once the scheduling is done we remove the undefined tale.
  + There is also an undefined counter. If the counter is equal to the number of writers, that would mean we are done publishing all the tales and there aren’t any tales present to be published.
  + There is a *limitDaily* counter that makes sure only 10 tales are being published every day. When the counter reaches 10 then we increment *talePublishDate* and set limitDaily to 0 again.
  + If the talePublishDate = totalScheduleDays, i.e. 30 as we are considering for a month, then the loop stops and we have our schedule ready.
* ***keyValuePairPublishedWriters()*** – This is similar to keyValuePairWriter() function. Here we take all the distinct writers with only the tales that has been published.
* ***setHappinessIndex()*** – Here we set the happiness Index of the writer also the income earned. Happiness Index is set by ***(Number of Tales Published/Total Number of Tales Submitted) \* 10.*** Income is set by ***Total Number of Tales Published \* 1000*** as the writer get Rs 1000 for every tale published.

Considerations during designing,

* As all the tales that have been selected are of equal quality, tales have been segregated according to the writers and then published one by one.
* As the algorithm must strive to minimize writers leaving TTT, it is made sure that none of the writers earn nothing in a month provided that the number of writers are equal to or less than the tales limit per day \* total number of days in the schedule (10 \* 30 = 300).
* To minimize the overall unhappiness, considering the above, every writer would go public at least once.
* To increase the overall happiness, all the writers who have submitted a decent amount of tales, a fair amount of tales of each writer has been published.
* Those who have submitted an exceeding amount of tales, it is really appreciated but then submitting a higher number of their tales would decrease individual happiness of many and would make them leave TTT. Hence to be fair to the high numbers, the writers have been sorted in descending order of the tales submitted i.e. writers with more number of tales would have a slightly higher preference and are first in the queue.
* Happiness Index can be within a range of 0 to 10, where 0 being least happy and 10 being extremely happy. It is defined by ***(Number of Tales Published/Total Number of Tales Submitted) \* 10.***
* According to the data provided, we have 74 writers.
  + The overall Happiness Index attained using this algorithm is 613/740.
  + A total of 10 writers have their Happiness Index lesser than 5.
  + A total of 64 writers have their Happiness Index above 5.
  + 50+ writers have their Happiness Index 8 or above.