# FIT3155 - Assignment 1

Name - Sadeeptha Bandara
ID - 30769140

## Comments to the marker: Q2

The implementation for this question is incomplete since I found myself running short on time.
The boyer moore implementation contains all of the standard bad character shifts, good suffix and match prefix shifts as well as galil's optimization. It should work with no bug for all patterns without a wildcard.

The bad character matrix is computed successfully taking into account the wildcard character. Runs in O(m) time since there is only at most one wildcard in the pattern.

The good suffix and match prefix arrays are computed in a single function in order to save an additional O(N) time complexity. They make use of a modified z suffix output taking the wildcard into account.

The z suffix function computes the longest substring ending at each index that matches the suffix and considers the wildcard in it's computation. The gs and matched prefix makes use of this array and would therefore consider the wildcard.
However, it is possible that a couple of edgecases have been missed.

The standard boyer moore implementation has not been modified to accommodate logic for the wildcard shifts, though it makes use of the computed z suffix and bad character matrices.

However, given time my implementation would change as follows

1. If the wildcard was within the suffix region, an additional comparison would be made between the equivalent position of the rightmost good suffix if it exists
2. If the wildcard to the right of the suffix region, it can be shifted safely as the wildcard would match

There would be similar variations of the above rules for the match prefix case.

Overall implementation is in the order of O(m+n) time and O(m) space, taking up more constant time space than the standard implementations in order to accommodate the wildcard.