**Name: Lam Kim Ze**

**Student ID: 31860346**

## Documentation Q2

To solve this question, I am using Boyer Moore with Galil optimization. The approach that I had done is creating two more table for good suffix and matched prefix instead of two array. In the two table, we considered every possibility on the wildcard which is length of 26 (a to z) and calculate the good suffix and matched prefix of each of them. The preprocess of bad character table is just adding an additional line for wildcard. Then, the preprocesses are mostly done. Regarding the shifting, we compared from the back to front and before start comparing, we check whether the character we had compared before or not. If yes, we just skipped it which known as Galil's Optimization. Then, we find the position of wildcard along comparing. If we found the wildcard, we noted down the character from the text that refer to the wildcard. Then, we focus on that line of matched prefix and good suffix. We then choose the greatest shift among bad character shift, good suffix and matched prefix when we found a mismatch and there is no wildcard exist. The pointers of Galil's optimization are updated here. There is a flag that turn True here and we exit the for loop. If the flag is False means matched with pattern then we append to the result.