Jacobs University Bremen

Assignment #1

Student name: *Ivan Kabadzhov*

Course: *Introduction to Computer Science*
Date: *September, 2018*

Problem 1.1: boyer moore bad character rule

a) When a character is compared it is indicated with a capital letter. If it is a match, it is coloured in green; if it is a mismatch it is in red. We simply move the pattern in the text from left to right without skipping and compare the characters from left to right as well.

A	B	A	A	B	D	A	B	B	A	B	A	B	B	C	A	B	A	B	A
🟥 B	a	b	a																
	🟩 B	🟩 A	🟥 B	a															
		🟥 B	a	b	a														
			🟥 B	a	b	a													
				🟩 B	🟥 A	b	a												
					🟥 B	a	b	a											
						🟥 B	a	b	a										
							🟩 B	🟥 A	b	a									
								🟩 B	🟩 A	🟩 B	🟩 A								

Number of alignments: 9
Number of comparisons: 16

b) When a character is compared it is indicated with a capital letter. If it is a match, it is coloured in green, if it is a mismatch it is in red. We move the pattern from left to right, but we compare the characters from right to left. And we are searching for the occurrence of the mismatch from the pattern in the text and we move the pattern accordingly. Because the pattern does not have many diverse letters we do not make a lot of skips.

A	B	A	A	B	D	A	B	B	A	B	A	B	B	C	A	B	A	B	A
b	a	🟥 B	🟩 A																
	b	a	b	🟥 A															
		b	a	b	🟥 A														
						b	🟥 A	🟩 B	🟩 A										
							b	a	b	🟥 A									
								🟩 B	🟩 A	🟩 B	🟩 A								

Number of alignments: 6
Number of comparisons: 12
Number of skips: 3

c) We count the position of the characters starting from 0: $B = 0$; $A = 1$; $B = 2$; $A = 3$. This is a preparation for the bad character rule to find the occurrence of the pattern in the text. The table indexes by character in the alphabet not matching this character of the pattern and the position of the character in the pattern.

	B	A	B	A
A	0	-	0	-
B	-	0	-	0
C	0	1	2	3
D	0	1	2	3

d) Every last occurrence of the character in the pattern is stored and if the alphabet character does not exist in the pattern then we write -1. We compare our data with **b)**.

A	B	C	D
3	2	-1	-1

A	B	A	A	B	D	A	B	B	A	B	A	B	B	C	A	B	A	B	A
b	a	B	A																
	b	a	b	A															
		b	a	b	A														
						b	A	B	A										
							b	a	b	A									
								B	A	B	A								

Problem 1.2: leap year in the Gregorian calendar (haskell)

```haskell
isLeapYear::Integer->Bool
isLeapYear ily =
  (mod ily 4 == 0) && not (mod ily 100 == 0) || (mod ily 400 == 0)
-- we consider everystep following AND and OR
-- AND has precedence over OR

isLeapYear'::Int->String
isLeapYear' ily'
  | mod ily' 400 == 0 = "It is a Leap year"
  | mod ily' 100 == 0 = "It is not a Leap year"
  | mod ily' 4 == 0 = "It is a Leap year"
  | otherwise = "It is not a Leap year"
-- we have to consider the order because the first thing has precendence
-- so we start from the cases involving least numbers (divisible by 400)
```