JOHANNES KEPLER
UNIVERSITÄT LINZ

Algorithms and Data Structures 2
Winter term 2019
Stefan Grünberger, Dari Trendafilov

Assignment 06

Deadline: **Mon. 16.12.2019, 23:59**
Submission via: **www.pervasive.jku.at/Teaching/**

Elaboration time

*Remember the time you need for the elaboration of this assignment and document it in the file **time.txt** according to the structure illustrated in the right box. Please do not pack this file into an archive, but upload it as a **separate file**.*

```
#Student ID
k12345678
#Assignment number
06
#Time in minutes
190
```

Strings & Pattern Matching

Use the class **Result** for the return values in this assignment. Objects of this class contain the number of found occurrences of the pattern in the text, as well as all start indices (in the string) of the found occurrences. A null string in the pattern or text should trigger an *IllegalArgumentException*. An empty string in the pattern should raise an *IllegalArgumentException* as well. In case that

- the text is empty, or
- the pattern is longer than the text, or
- no result is found,

an object of type *Result* with "0 matches" and an empty *resultIndices* array should be returned. Use the following implementation:

```
public class Result {
        Integer resultIndices[];
        Integer count;

        public Result(Integer resultIndices[], int count) {
                this.resultIndices = resultIndices.clone();
                this.count = new Integer(count);
        }
}
```

1. KMP algorithm — 4 points

Implement the KMP search algorithm as demonstrated in the exercise. The algorithm should return an object of the class *Result*. Use the following interface:

```
public class KMP {
  static public Result search (String pattern, String text) throws IllegalArgumentException {…}
}
```

Example:
For the sequence „*abcdexxxunbxxxxke*" and the pattern „xxx" the algorithm should return the following result:

Internal String representation:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
a	b	c	d	e	x	x	x	u	n	b	x	x	x	x	k	e

Result:

```
Result([5,11,12], 3)
```

JOHANNES KEPLER
UNIVERSITÄT LINZ

Algorithms and Data Structures 2
Winter term 2019
Stefan Grünberger, Dari Trendafilov

Assignment 06

Deadline: **Mon. 16.12.2019, 23:59**
Submission via: **www.pervasive.jku.at/Teaching/**

2. Rabin-Karp algorithm — 8 points

Implement the **Rabin-Karp** search algorithm as presented in the exercise. A hash value is calculated for the pattern (of length m), and for the sequence from the text with the same length. If both hash codes are identical, the brute-foce method is used to continue with a character by character comparison.
Consider, it is essential to use the **rolling-hash function** for the calculation of the hash values. The algorithm should return an object of the class *Resut*.

Example:
For the sequence „*abcdexxxunbxxxxke*" and the pattern „xxx" the following result is expected:

Internal String representation:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
a	b	c	d	e	x	x	x	u	n	b	x	x	x	x	k	e

Result:

```
Result([5,11,12], 3)
```

Use the following interface/class for your implementation:

```
public class RabinKarp {
        static public Result search (String pattern, String text) throws
IllegalArgumentException {…}

        /**********************
        * auxiliary methods
        **********************/

        // return the hash value of a character
        static int getHashValue (char[] c) {…}

        // used for Integer Arithmetic
        static int pow (int base, int exp) {…}

        // if necessary, define other private auxiliary methods
        …
}
```

Hint:
We suggest to use the character coding according to the ASCII table.

3. Boyer-Moore algorithm — 12 points

Implement the **Boyer-Moore** search algorithm. It should return an object of the class result.

Example:
For the sequence „*abcdexxxunbxxxxke*" and the pattern „xxx" the algorithm should return the following result:

Internal String representation:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
a	b	c	d	e	x	x	x	u	n	b	x	x	x	x	k	e

Result:

```
Result([5,11,12], 3)
```

JOHANNES KEPLER
UNIVERSITÄT LINZ

Algorithms and Data Structures 2
Winter term 2019
Stefan Grünberger, Dari Trendafilov

Assignment 06

Deadline: **Mon. 16.12.2019, 23:59**
Submission via: **www.pervasive.jku.at/Teaching/**

Use the following interface/class for your implementation:

```
public class BoyerMoore {
        static public Result search (String pattern, String text) throws
IllegalArgumentException {…}

        /**********************
        * auxiliary methods
        **********************/

        //
        // --> see the exercise slide set for explanations

        // is pattern[0:m-s-1] a suffix of pattern[k=i+1:m-1]?
        static boolean checkIsSuffix(char[] pattern, int k) {…}

        // length of longest string in pattern[0:m-s-1] that is a suffix
        // of pattern[k=i+1:m-1]
        static int getLongestSuffixLength(char[] pattern, int k) {…}

        // if necessary, define other private auxiliary methods
        …
}
```

Hint:
For the array `last` you can assume a fixed size of 128 (ASCII alphabet).