Lab 3

Reading

M. T. Goodrich, R. Tamassia and M. H. Goldwasser, *Data Structures and Algorithms in Java*, 6th Edition, 2014.

- Chapter 3. Arrays and Linked Lists
- Chapter 6. Stacks and Queues

Exercise 1

- Implement the Stack ADT using an array.
- Implement the Stack ADT using a linked list.

Exercise 2

- Implement the Queue ADT using an array.
- Implement the Queue ADT using a linked list.

Exercise 3

- Write a Java program to remove duplicates from a given stack.
- Write a Java program to finds common elements between two stacks.

Exercise 4

Write a Java program to reverse a queue.

Exercise 5

Design a method to check whether delimiting symbols match up correctly, and implement it in Java. Examples of correct and incorrect groups of delimiting symbols are provided below. You may want to use them to test your program.

- Correct: ()(()){([()])}
- Correct: ((()(()){([()])}))
- Incorrect:)(()){([()])}
- Incorrect: ({[])}
- Incorrect: (

Exercise 6

In the children's game "hot potato", a group of *n* children sit in a circle passing an object, called the "potato" around the circle. The potato begins with a starting child in the circle, and the children continue passing the potato until a leader rings a bell, at which point the child holding the potato must leave the game after handing the potato to the next child in the circle. After the selected child leaves, the other children close up the circle. This process is then continued until there is only one child remaining, who is declared the winner. If the leader always uses the strategy of ringing the bell so that every *k*th person is removed from the circle, for some fixed value *k*, then determining the winner for a given list of children is known as the ***Josephus problem*** (named after an ancient story with far more severe consequences than in the children's game). Design a method to solve the Josephus problem, and implement it in Java.

Exercise 7

Transform the algebraic expression with brackets into RPN form (Reverse Polish Notation). Two-argument operators: +, -, *, /, ^ (priority from the lowest to the highest), brackets ( ). Operands: only letters: a, b,..., z. Assume that there is only one RPN form (no expressions like a*b*c). More detailed descriptions are here.