Kristianstad University
SE-291 88 Kristianstad
Sweden
+46 44 250 30 00
www.hkr.se

DA256D:
Algorithms and Data Structures

Daniel Einarson & Kamilla Klonowska

Seminar 2
Lists, Stacks, Queues

Goals

The goal of this seminar is to prepare the students to fulfill the learning outcomes

Skills and abilities

After completing the course students must

- be able to apply the algorithm theory and data structures in practice (3)
- be able to seek, gather and critically interpret relevant information to formulate responses to well defined issues in algorithm theory (4)
- be able to describe and discuss its own expertise with various student groups (5)
- be able to work independently and in groups (6)
- be able to formulate a relevant research question in algorithm theory and data structures (7).

Before the Seminar –

The seminar is mandatory. A student that did not upload the report and the programming tasks before the seminar is not allowed to participate in the seminar.

All tasks must be solved individually.

It is not sufficient to only solve the tasks, the student must also be able to explain the source code, explain how each algorithm and method works, discuss the method of evaluation of the algorithms and data structures, as well as present the results in scientific way. Otherwise the tasks will not be approved. Note that hints of how some of the different algorithms work or how to implement them can be found in the course literature [1]. Make sure that for all other solutions that you find on the Internet and used as inspiration you have to refer to the source. Make sure to understand the algorithms before implementing them.

Task 0

Read the article "Verifying Properties of Well-Founded Linked Lists" and write a short reflection.

Task 1

Implement the following exercise from the course literature: [1] Ex.3.21, p.119.

Write a program in your chosen programming language to check for balancing symbols in the following languages:

a. for language C: (/* */, //, [ ], { }).

b. For language C++: (//, ( ), [ ], { }).

c. Explain how to print out an error message that is likely to reflect the probable cause.

d. Write a reflection: Which data structure is most suitable to solve this problem and why? Explain your answer.

Note: you are not to ask to program in C or C++. Your task is to implement a program that works as a compiler to check syntax in these programming languages.

Examples

The following examples can be used to test the functionality of a "balance checking program". Each statement shall be entered as one String (not character by character).
There are of course many other examples, some more and some less tricky but make sure the application solving Task 1 for Seminar 2 will at least pass the following tests.
Worth to note is that the // signs should comment out the rest of the line, that is until it reaches a line break. The student should reflect upon how the /* */ comments work and especially how /* works when there is no */ in the supplied String.

For Task 1a	For Task 1b

Valid:
1. int a;
2. int a; /* random comment */
3. int a; /* for storing width * height */
4. int a = b*c;
5. int a = b / c;
6. int a = 55; // This is a comment / [
7. public void doIt(int x) {System.out.println(x*100);}
8. int []arr = new int[10];
9. /* */ {}
Not valid:
1. int [arr = new int[10];
2. int b = 5; /* this is a comment /
3. {a=b;
4. }
	Valid:
	1. if(a == b) {a++;}
	2. if(a < (b*c)) {t = 5;}
	3. int []b = new int[5];
	4. []{}
	5. int a = 5; // init a to 5
	Not valid:
	1. for(int i=0;i<10;i++] {a+= b;}
	2. {abc)

Task 2

Implement the following exercise from the course literature: [1] Ex.3.24, p.119.

a. Write routines to implement a queue using two stacks.
b. Write routines to implement a queue using only one stacks.
c. Write routines to implement a stack using two queues.
d. Write routines to implement a stack using a single queue.

Your routines should have a warning message for underflow and overflow.
Write a reflection: Which routines are most efficient? Explain!

Task 3

Implement an address book using a LinkedList to store the contacts.
Each contact shall be represented by one node in the linked list and shall store the name and address of the contact. The linked list must be implemented by the student manually, i.e. it is not allowed to use Javas build in implementations.
The linked list must have at least the following functionality:
1) Add node
2) Remove node
3) Get node using index
The program must use all of the above functionality and must print the complete list of contacts to the screen using a loop.

Task 4

Implement the following exercise from the course literature: [1] Ex.3.6, p.116.

The Josephus problem is the following game: N people, numbered 1 to N, are sitting in a circle. Starting at person 1, a hot potato is passed. After M passes, the person holding the hot potato is eliminated, the circle closes ranks, and the game continues with the person who was sitting after the eliminated person picking up the hot potato. The last remaining person wins. Thus, if M = 0 and N = 5, players are eliminated in order and player 5 wins. If M = 1 and N = 5, the order of elimination is 2, 4, 1, 5.

a. Write a program to solve the Josephus problem for general values of M and N. Try to make your program as efficient as possible. Make sure you dispose of cells.

   Implement the program using three methods:

   - one with **ArrayList**
   - one using ArrayList with **Iterator**

- one with your own **LinkedList**
- one using Linked list with **Iterator**

b. Compare the running time of your program depending on both data structures. Present the results in a table and a diagram (plot). Think about different input sizes! Can you find a mathematical formula to predict the output for any input?

c. Write a reflection: When should **LinkedList** be used over **ArrayList** and vice-versa?

Note: your program should work for **any input**.

During the Seminar –

Each student is responsible for one (random) Task for the examination. The student is responsible for the task, i.e. explains the solution, presents a code, discusses the complexity / efficiency of the code/algorithm, discusses other students' solutions, as well as is responsible for the analysis of the results of the current task.

References

1. Weiss, Mark. A. (2012), Data structures and algorithm analysis in Java. 3rd edition Harlow, Essex : Pearson. (632 p).