- Problems are **due by Wednesday, March 14th before class**. Remember that groups consist of **3 persons**!!
- Programming problems (marked with "**P**") have to be implemented and a **printout** of (i) the code including documentation, and (ii) meaningful test outputs are to be submitted.

Group Assignment 3

Problem 1 (P, Java, ADT Queue) Implement the abstract data type *Queue* from the previous group assignment in Java. You should consider using `Javadoc` for documentation purposes (recommended but not required).

a) The *specification* of the ADT *Queue* should include the Java `interface` declarations of all queue operations, together with the *constraints* of the ADT (e.g., as mathematical equations and/or clear unambiguous English).

b) Develop an implementation `ArrayQueue` of the ADT based on arrays as a concrete type.

c) Develop one implementation `ListQueue` of the ADT based on lists as a concrete type.

d) Compare your different implementations, including the earlier Haskell implementation (provide the (possibly updated) Haskell code as part of the current assignment!). Some points to consider are: programming effort, clarity of code/maintainability, efficiency, limitations (polymorphism/element types, queue size, ...), etc.

Problem 2 (Design/specification issues of PLs) Consider the following constructs of a "mini" programming language `mPL`:

- a *procedure* has a *head* and a *body*
- *head* consists of a *procedureName*, *formalParams*, *variableDeclarations*, and *localProcedures*
- *formalParams* is a list of pairs $(paramName, paramType)$
- *variableDeclarations* is a list of tuples having a *varName*, *varType*, and optionally *varInitialValue*
- *body* is a sequence (=list) of *statements*
- a *statement* is an *assignment*, a *procedureCall*, or an *ifThenElse*
- an *assignment* has a *varName* as the lhs and an *expression* as the rhs
- an *expression* is an arithmetic expression involving "+", "–", "*", and "/" that is built over *varNames* and *literals*
- a *procedureCall* consists of a *procedureName* and *actualParams*
- *actualParams* is a list of *varNames* and *literals*
- an *ifThenElse* comprises a *booleanCondition*, and two *statements* (then+else parts)
- a *program* is a *procedure* whose name is `main`

a) Give an *abstract syntax notation*[1] (e.g., as Haskell data types) of `mPL`.

   To do so, first complete/disambiguate/rectify the above partial specification where necessary and justify your choices.

b) After having fixed the abstract *syntax* in part (a), explain what needs to be done in order to fix a precise *semantics* of `mPL` (in other words, what crucial semantic aspects are lacking, even considering an intuitive/common understanding of the terms used `mPL`).

c) Make some choices for the semantics (scoping rules, parameter passing, ...) and sketch how you could implement the resulting (now completely specified) language `mPL` in (i) Haskell, and (ii) Java.

[1] not to be confused with ADT!