# Laboration 1

Due 21118.

Let *bSort* be a sorting algorithm that is a variant of *Insertionsort*, where the linear search technique (which is used for locating the position in which to insert the new item) is replaced by a binary search technique.

Consider the following two modifications to *Mergesort*: First, the input is divided into $\frac{n}{k}$ sublists each of length $k$, where $k < n$ is a value to be determined and $n$ is the length of the input list. Then each sublist is sorted using *insertionsort* and *bSort*, respectively. After that, the sorted sublists are merged using the standard merging mechanism. Your task is to implement and evaluate these two modified mergesorts.

- Implementation and experiment

  Implement the algorithms (using either *Python* or *Java*) and investigate their performances by testing your code on *large* data sets. The time complexity of the algorithms can be measured by both the number of main operations performed and the total running time (excluding the one needed to generate the test data) taken by the algorithms. The algorithmic efficiency testing *may* be carried out as follows:

  - Conduct simulation experiments with various input settings (for example, random data, already sorted data, almost sorted data, ... ), while changing the value of $k$ and the input size.
  - Measure and illustrate the algorithm performances against each other, against several different criteria (for example, time and space requirements of a program, *and so on*), and against the standard mergesort.
  - Estimate the best values of $k$.

- Report

  Each group must turn in a report describing and illustrating experimental results. The report can be written in either Swedish or English and should not be handwritten.

  *Before submitting your report, you should discuss your solution to the laboration (design, implementation, test, and report) with your lab-assistants.*