Algorithm Engineering
Funke/Koch/Weitbrecht SS 2025
http://www.fmi.informatik.uni-stuttgart.de/alg

Institut für **Formale**
Methoden der Informatik
Universität Stuttgart

Assignment Sheet 3

Problem 1

Implement the following three strategies for intersecting two sorted arrays/vectors $A[]$ and $B[]$ of numbers:

- naive intersection (like mergesort)
- binary search within $A[]$ for each element in $B[]$
- galopping search within $A[]$ for the elements of $B[]$

Write a program which:

- takes as command line argument the number n of elements in $A[]$
- initializes $A[]$ with the numbers $0, \ldots, n-1$
- for $i = 0, 1, \ldots, \lfloor \log_2 n \rfloor$:
  - initialize B with $n/2^i$ random numbers in $[0, n-1]$ (sorted in increasing order and cleaned of duplicate numbers)
  - intersect $A[]$ with $B[]$ using the three strategies and measure the running time

Additional remarks/questions:

- on a machine with 16GB of RAM your code must be able to intersect vectors of size $|A| = |B| = 10^8$ in at most 2 seconds
- compare running times of the three strategies; at what sizes is binary/galopping search the preferred method?

Problem 2

On the Ilias course page you can finde a file `movies.txt.bz2` containing movie descriptions (title followed by plot summary – all in one line per movie).

Implement a 'movie search engine' based on an inverted index, such that it is possible to query the data set by specifying a sequence of words. As a result all movies whose description contains all words given in the query are presented.

- implement two variants of the inverted index – one based on a hashmap, one based on a sorted sequence (e.g., search tree)
- output construction time and query time for both variants (use one of the intersection routines from the previous problem solution)

- additionally compare the query time to the (very) naive solution, where each movie description is checked for presence of all query words.
- make your 'search engine' more usable by unifying upper/lower case and presenting results in a reasonable order, e.g., by using inverse document frequencies, or preferring movies where the search words appear in the title

Output the construction time and the query time (use one of the intersection routines from the previous problem solution).

Problem 3
Submit solutions to *both* problems in Ilias with a Makefile and a README file explaining how to compile/execute your implementations.