# CSCI 2100: Project

Prepared by Yufei Tao

This project should be completed by each student individually. The objective is to implement a *static* binary search tree (BST), and more specifically: an algorithm for building a BST efficiently and an algorithm for using the BST to answer predecessor queries.

**Data and Queries.** You will be given a file named "ds.txt", which describes a set $S$ of $n$ integers in the format of one integer per line. For example, the following lines describe the set $S = \{25, 81, 47, 23, 59\}$ with $n = 5$:

25
81
47
23
59

You will be given another file named "qry.txt", also with $n$ lines, where each line has the form "`qry` $q$", with $q$ being the search value of a predecessor query. For example, the following file describes $n = 5$ queries with search values 95, 23, 7, 57, 72, respectively:

`qry` 95
`qry` 23
`qry` 7
`qry` 57
`qry` 72

Your program should build a BST on $S$ using the sorting-based algorithm explained in the Week 11 tutorial. Then, for each predecessor query, you should write the answer to a file named "**output.txt**" (if the predecessor does not exist, write "no"). For example, after processing the aforementioned queries, your file should have 5 lines:

81
23
no
47
59

The "ds.txt" and "qry.txt" files used to test your program will contain $n$ = 1,000,000 integers and 1,000,000 queries. The first half of the data and query files will be made available for download on the course homepage, whereas the other half will be hidden from you.

**Programming Language.** C++ (including variants like C, C#, ...), Java, or Python. Your implementation must be from *scratch*, namely, you can use **only** functions from a standard library, some examples of which are shown below:

- C++: A standard MSDN library (docs.microsoft.com/en-us/cpp/?view=msvc-160) or GNU library (www.gnu.org/software/libc).
- Java: A standard edition (e.g., docs.oracle.com/javase/8/docs/api).
- Python: The library at docs.python.org/3/library.

**Grading Scheme.** Your score will be calculated as follows:

- 0 marks, if your source code cannot be compiled.
- 0 marks, if you did not implement the construction algorithm in the Week 11 tutorial.
- If your program runs, we will obtain the number $m$ of queries that your program answers correctly. Then:
  - (C++/Java) If your program finishes within 10 seconds—this time covers *everything*, including file reading and writing—then your final score will be $m/(10000)$. If your program finishes within 20 seconds, your final score will be $m/(20000)$. If your program takes more than 20 seconds, 0 marks.
  - (Python) Replace the numbers 10 and 20 in the above bullet with 50 and 100, respectively.