This assignment is marked out of 100 and is worth 7.5% of your course grade.

1. Minimum subarray sums (50 marks)

Devise an algorithm and implement it in a program to solve the following problem. For an input array $a[0..n-1]$ of size n (where each element is an integer), and an integer m with $1 \leq m \leq n$, a *subarray sum of size* m is any sum of m consecutive elements of the array, namely $\sum_{i=j}^{j+m-1} a[i]$, where $0 \leq j \leq n-1$ and $j+m \leq n$. Your program should take a and m as input, and output the minimum value over all subarray sums of size m.

- You should first make sure that your algorithm and implementation are correct! It is up to you to come up with test cases and the corresponding correct output. Usually, if a program contains an error, this can be detected on a small test case. You are allowed to share test cases with the class via the class forum.
- Next you should consider efficiency. Remember that the test cases may use any value of m and n.
- Please see below for information on input and output formats.

2. Top songs (50 marks)

Devise an algorithm and implement it in a program to solve the following problem, similar to one often faced by an MP3 player. For our purposes, a song consists of the following data fields: title (a nonempty ASCII string), composer (a (possibly empty) ASCII string), running time (a positive integer). Input consists of n songs and an integer k with $1 \leq k \leq n$. Your program must find the k songs with longest running times, and output these songs in descending order of length of song. If songs have the same running time, then we use lexicographic order on the title, and then on the composer, to get a total ordering. The lexicographic order comes from the usual order on ASCII characters.

- You should first make sure that your algorithm and implementation are correct! It is up to you to come up with test cases and the corresponding correct output. Usually, if a program contains an error, this can be detected on a small test case. You are allowed to share test cases with the class via the class forum.
- Next you should consider efficiency. Remember that the test cases may use any value of k and n.
- Please see below for information on input and output formats.

Questions involving programming

- Python, Java, C++ and some other languages are supported.
- Your answer to each question should be a single file (containing all nonstandard classes you use). You may assume that the markers have access to all standard libraries.
- A sample input and output file for each question will be available from Canvas. The markers may check the output with a text comparison program, so it must be in EXACTLY the right format.
- The automated feedback and submission system ("automarker") is available. You must submit your answer via this system. You may take account of the feedback given by the automarker, and resubmit before the deadline without penalty.
- Your program(s) may be tested on randomly generated input files, some of which may be very large. Some of these test files may not be those used as feedback test cases on the automarker. Marks will be allocated for correctness and speed of the programs (usually about 60% for correctness, 40% for speed). Simply "passing" the test cases on the automarker may not guarantee maximum marks, but it will guarantee more than half marks for correctness. If full marks for correctness are not obtained, then the marks for speed are automatically set to zero.
- No marks will be awarded for comments, but you must at least include comments with the name of the author, UPI, and the purpose of the code.

Input and output format

For this assignment, the following format will be used. Sample input and output is available on Canvas. Pay attention to line breaks and beware of nonstandard software such as anything made by Microsoft. For best results, use a Linux/Unix environment (the automarker does).

You can assume that input will come from standard input in a stream that represents one string per line. Output should be sent to standard output. In the automarker, the input will come from a file that is piped to standard in and the output redirected to a file, but your program shouldn't know that.

For Q1, the first line gives n and m. The next n lines contain $a[0], \ldots, a[n-1]$ respectively, one element per line.

For Q2, the first line gives the integer k, and the second line a separator character that is not contained in any of the song or composer strings. The subsequent lines give the song data, with fields separated by the separator character.