

We think problem solving is one of the most important assets of a software developer. Please read through the tasks below and try to propose your solutions.

The tasks from the first section are quite simple and would not take long time to find a good solution. The last task may require about 2-3 hours and extensive knowledge of various aspects of programming. And this is an example of a simple task that may happen every day at Eloquera.

Algorithmic tasks

1. Given are two SZ-arrays (SZ = single-dimensional, zero-based) of integers (*Int32*) with unique elements ordered in an increasing manner. Find the intersection of these two arrays (*intersection* here should be treated in terms of a set theory, i.e., the elements appearing in both arrays).
2. Having the arrays from the first task, find the union of these two arrays.
3. Estimate the number of operations/time required for executing two previous operations (using big-O notation)
4. You have a linked list (unidirectional or *one-way* list). Suggest an algorithm telling if the linked list has loops in it.
5. Propose a way of finding the *naughty* link from the linked list with the loop.
6. When you open a hot water tap at home, you can notice that the first few seconds after opening a tap, the water is still cold. It can be easily explained by the cooling the stationary water in the pipes. Although at some good hotels when you open the hot water tap, the water is instantly hot. Please give the explanation of this fact. You are not bound by any number of explanations and can give as many versions as you like.

Please do not use any standard containers or methods (like `LIST<T>` or `INTERSECT()`) for solving the tasks above. All these standard ways of solving the problems are far from being optimal.

Parallel calculations task

Create a simple console application that calculates the Mandelbrot set on multiple machines.

From an input file the application gets:

- Names or IP addresses of the machines which it should use for calculation
- Step of increment for calculation (e.g., 10^{-6})

The application should automatically split the calculation task between multiple machines, and be able to pause calculations if required and save the intermediate results into a file. The file can be used to continue calculations without starting them all over.

The pause command is issued by running another instance of the application with */stop* command line parameter. The second instance finishes immediately, and the main instance stops all calculations, saves all calculated data including the range to be calculated into a file.

Starting the application with the file name as a command line parameter shall resume calculations.

The Mandelbrot set is calculated using the following rule:

if for a given complex value c

$$z_n = z_{n-1}^2 - c,$$

$$z_0 = 0$$

the sequence z_n converges, then the complex value c is included into the resulting Mandelbrot set.

Tip: If at some moment $|z_n| > 2$, then the sequence for a given c diverges. You can limit $n < 100$.

Please do not use any third-party parallelization libraries or technologies like MPI. Write your C# code clearly, optimize for performance.

Send your answers along with your resume to job002331@eloquera.com

Good luck!