Given integers $\boldsymbol{x} = x_0, x_1, \ldots, x_{n-1}$ we seek to generate the numbers $S_0, \ldots, S_{n-1}$ where

$$S_i := \sum_{j=0}^{i} x_j.$$

In what follows a pseudo-parallel algorithm for this problem is outlined. It should be noted that numerous parallel and pure parallel algorithms for this problem are in existence as this is a classical (applicative) CS problem. We chose to describe what is perhaps the simplest of them all. We will use the figure below in order to illustrate this algorithm.

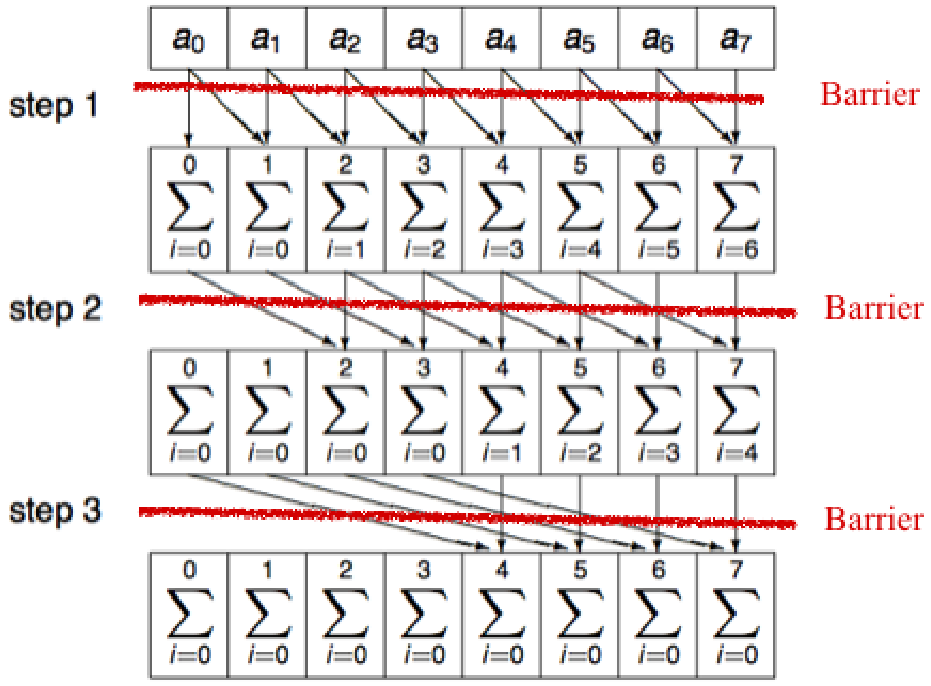

Figure 1: Illustration of the parallel-prefix algorithm

We employ n processes and one array of size n denoted A. We assign cell i in the array to process i and throughout it always writes to cell i and only to cell i. It could however read from other cells of this array.

Initially array A is loaded with the original series $\boldsymbol{x}$. The algorithm works in "waves" or "steps" where in order to begin each step one first needs to align all (remaining) participating processes using barrier synchronisation.

1. Step 1: In this step process $i \in [1, n-1]$ reads cells $A[i-1]$ and $A[i]$ sums them up and writes the solution to $A[i]$. Process 0 has to do nothing (and actually it is not needed at all throughout the algorithm).

    - At this moment you note that processes 0 and 1 are done and S_0 and S_1 now can be found in $A[0]$ and $A[1]$.
    - Note also that it is crucial that all processes first finish reading the two values each of them needs and only after this is done they can start writing into A their new values. Hence the first barrier is drawn in the picture.

2. Step 2: The second step is essentially identical to the first only that now the places from which the processes read in A changes. Moreover, processes 0 and 1 are no longer needed.

    Process $i \in [2, n-1]$ reads $A[i]$ and $A[i-2]$ (see picture) and writes this to $A[i]$.

    - At this moment you note that processes 2 and 3 are also done and that S_2 and S_3 now can be found in $A[2]$ and $A[3]$.
    - Note also that again it is crucial that all processes first finish reading the two values each of them needs and only after this is done they can start writing into A their new values. Hence the second barrier is drawn in the picture.

3. The algorithm continues in this way until all S_i's have been generated.

Submit the following to the TAs Moodle.

A Assume n is even. Implement the above algorithm using C (over Linux).

B Assume n is even. Implement the above algorithm using Java (with threads now).