## Threaded Implementation of Berkowitz Algorithm (Part A)

In this assignment, you are going to implement Berkowitz's parallel algorithm for computing the characteristic polynomial of a matrix. For the sake of simplicity we assume that the matrices are over $\{0, 1\}$, i.e., the field of two elements, where plus is XOR and multiplication is AND.

The coefficients of the char poly of an $n \times n$ matrix $A$ below, are computed in terms of the coefficients of the char poly of $M$:

$$A = \begin{bmatrix} a_{11} & R \\ S & M \end{bmatrix}, \tag{1}$$

where $R, S$ and $M$ are $1 \times (n-1)$, $(n-1) \times 1$ and $(n-1) \times (n-1)$ sub-matrices, respectively.

In matrix form: the char poly $p$ of $A$ can be expressed in terms of the char poly $q$ of $M$:

$$p = C_1 q, \tag{2}$$

where $C_1$ is an $(n+1) \times n$ Toeplitz lower triangular matrix, and where the entries in the first column are defined as follows:

$$c_{i1} = \begin{cases} 1 & \text{if } i = 1 \\ -a_{11} & \text{if } i = 2 \\ -(RM^{i-3}S) & \text{if } i \geq 3 \end{cases}. \tag{3}$$

For example, if $A$ is a $4 \times 4$ matrix, then $p = C_1 q$ is given by:

$$\begin{bmatrix} p_4 \\ p_3 \\ p_2 \\ p_1 \\ p_0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -a_{11} & 1 & 0 & 0 \\ -RS & -a_{11} & 1 & 0 \\ -RMS & -RS & -a_{11} & 1 \\ -RM^2S & -RMS & -RS & -a_{11} \end{bmatrix} \begin{bmatrix} q_3 \\ q_2 \\ q_1 \\ q_0 \end{bmatrix}.$$

Berkowitz's algorithm consists in repeating this for $q$ (i.e., $q$ itself can be computed as $q = C_2 r$, where $r$ is the char poly of $M[1|1]$), and so on, eventually expressing $p$ as a product of matrices:

$$p = C_1 C_2 \cdots C_n.$$

Your implementation should work as follows:

- There should be two types of methods: one for computing powers of a matrix, and one for computing iterated matrix products. That is, one method for computing $M^i$ for any $M$ and any $i$, and one for computing $M_1 \cdot M_2 \cdot \ldots \cdot M_k$, for compatible matrices.
- On input $M$, the output of the powering method should be a list of matrices of the form $\{M^1, M^2, \ldots, M^n\}$, where $M$ is $n \times n$, and on input $\{M_1, M_2, \ldots, M_k\}$, the output of the iterated matrix product method should be their product (if it is defined).
- It is important to carefully divide the computation into phases. In the first phase, for example, the program computes all the powers of all the submatrices. In the second phase, it computes the entries of the $C_i$'s, and in the third phase it computes the product of the $C_i$'s. Synchronize the threads accordingly, and in the documentation show that your computation has at mose $O(\log^2 n)$ *sequentially-many* steps.