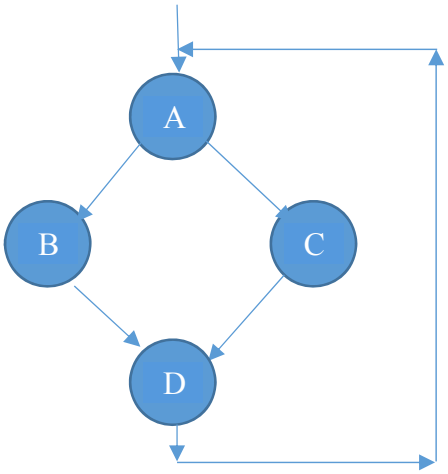
# Operating Systems

## *Lab 10 Exercise – Semaphores*

**Learning goals: this laboratory activity is devoted to the use of semaphores.**

## Exercise 1

Implements the following precedence graph with cyclic threads and POSIX semaphores. Each circle represent a thread created by the main thread, which terminates after the creation of these 4 threads. Each thread prints the corresponding character in the figure. Each thread loops 10 times, then it terminates.




## Exercise 2

The dot product of two vectors **a** = [$a_1$, $a_2$, ..., $a_n$] and **b** = [$b_1$, $b_2$, ..., $b_n$] is defined as:

$$\boldsymbol{a} \cdot \boldsymbol{b} = \sum_{i=1}^{n} a_i\, b_i = a_1\, b_1 + a_2\, b_2 + \quad \cdots + a_n\, b_n$$

where Σ denotes summation notation and *n* is the dimension of the vector space.

Write a C program that takes as arguments two numbers **`n`** and **`t`**.
The main thread creates **`t`** threads, and then it loops 10 times

- creating two vectors (arrays of **`n`** random numbers between 0 and 1, of type double)
- informing the threads that the new pair of arrays are available
- waiting for all the threads to complete their computations before
  - printing the dot product of the two vectors as computed by itself sequentially
  - printing the dot product of the two vectors as computed by the cooperating threads

Each thread

- waits that the main thread signal that the two vectors are ready
- performs the dot product of a different region of the two vectors
- updates, in mutual exclusion, a variable **`sum`**, which represents the partial dot product
- informs the main thread that it has completed its job
- until the main thread exit

Exercises 3 is in the next page.

## Exercise 3

Implements the following precedence graph with cyclic threads and POSIX semaphores. Each circle represent a thread created by the main thread, which terminates after the creation of these 6 threads. Each prints the corresponding character in the figure. Each thread loops 10 times, then it terminates. Please notice that **the small circle is not a thread**, just a synchronization point among threads B, C, D, and E.

Possible outputs:

**A B C D E F**

**A C B D E F**

**A C B E D F**

etc.