CS303 | Assignment-2 | Due 13/Oct/2021 11:59 PM | 100 points

- Important instructions for code submissions are here: https://goo.gl/IMWvdF
- Grading scheme to be followed is available here: https://goo.gl/52D82g
- Assignment description may be underspecified to allow some room for exploration and creativity.
- Your submission should be packaged as a zip file named **exactly** in this format: CS303-[your entry no.]-[assignment no.].zip.

We need to develop a program in C to simulate the following scenario involving detection of deadlocks among a pool of worker threads that will be started by this program. The simulation should determine whether a deadlock has occurred in a system of `n` identical resources and `m` threads. The program should take the following values as command line arguments:

1. Types of resources that are available on the OS and that can be requested by threads. Example: `A, B, C`, etc. Name doesn't really matter as long as it is unique.
2. Maximum number of instances available for each of the above resources. Example: `A=20, B=16, C=45`, and so on.
3. Maximum number of threads to use in the simulation.
4. Deadlock detection check interval `d` in seconds.

Each thread, `t`, should decide a random set R_t of resources that it wants (e.g., 4 instances of resource `A`, 3 of resource `B` and 1 of resource `C`). Then, the thread `t` will request the resources (from R_t) one type at a time in a random order and with random pauses between making requests for different resource types. The thread `t` should hold onto all the resources $R \subset R_t$ that it has acquired until it can get all of them, i.e., entire set R_t. Once a thread obtains all resources that it wanted, it should hold onto them for a random duration between (0.7d, 1.5d), then release them all at once. The thread should not terminate, but it should again decide a random set of resources to request, and repeat the process mentioned above.

The above pattern of working will lead to deadlocks among the worker threads. There should be a dedicated/separate thread `t'` to check for deadlocks among worker threads every `d` seconds. It should report when a deadlock has occurred and do the following:

1. Print the list of threads that are involved in the deadlock
2. Terminate the worker threads involved in the deadlock. This should be done by using suitable heuristics. For example, you may use the number of resources that are currently held by a thread to decide which thread to terminate first. The terminated threads should be replaced by new ones.
3. Track the time interval that is observed between the detected deadlocks.

You should use different heuristics for choosing worker threads to terminate and report which type of heuristic results in the longest and shortest average time between deadlocks. The simulation should continue to run until the user terminates it.