Blocking Operations:

I have implemented Dijkstra's approach to the dining philosopher problem, where out of k people only k-1 are allowed to sit on the table so that no condition for deadlock arrives (where one philosopher is holding one fork and waiting for the other). This deadlock resolution is done using the counting semaphore called room. As far as the forks are concerned, each of the k forks are represented using a binary semaphore and the function cSemWait is used to wait for a fork in a blocking way and the function cSemPost is used to signal that a fork is ready to be given away in a blocking way. Furthermore, the sauce bowls are also represented using a counting semaphore as they are multiple instances of same type. All the deadlocks are avoided and wait and signal takes place in critical section of each of the operation

Non-Blocking Operation:
Here the wait and signal operation returns a value, if the value is non zero all the resources that are possessed by the philosopher are also freed so as to determine that no philosopher holds onto their resources indefinitely, the signal used is also non blocking and keeps on running till the time resources are freed, furthermore the last portion of the thread has a series of blocking signals to indicate that the forks and bowls are free to use, I have used this blocking operation so as to make sure that it the wait has been successful the post is also successful and the philosopher doesn't hold onto the resources in a nondeterministic manner .