**Part 1**: In the given program, there are synchronization errors in which multiple processes are trying to access the same data (e.g., balance of bank) at the same time. To resolve this issue, we use binary semaphore or mutex lock which will allow the control of the dad process and his two children. What we first do is create the semaphore, in our case *semid*. Once we have done so, we manipulate its value through using two functions defined in *sem.h* - *P(semid)* and *V(semid).* The former is considered as wait (how long a process is to wait to enter the critical section) and the latter as signal (finished in critical section so release the lock). We place *P(semid)* at the beginning of each process and *V(semid)* at the end of each process. For example, when the dad makes a deposit, a wait is placed to prevent the sons from making any withdrawals and once done a signal is sent to the sons to let them know they can start withdrawing from the bank account.

**Part 2**: Now that we have resolved the synchronization errors, we can create a measurement component that allows us to know how long each process must wait, *T(P) -* is the number of times processes other than P enter the CS after P expresses interest in the CS but before P actually enters the CS. To implement this, we create three variables initialized to 0 to represent the wait time/number of processes for each process as well as three flags to represent each process interest. We incorporate that into our program placing it right after the *P(semid)* such that for the first process we increment the wait for son 1 and son 2. The second process, we increment the wait for dad and son 2 and the last process we increment dad 1 and son 1. In addition, we make sure that flag is not equal to o (i.e., should equal 1). We then run our program for 5 different N’s to measure the *T(P)* as shown below.

|  |  |  |  |
| --- | --- | --- | --- |
| N | Dad | Son 1 | Son 2 |
| 5 | 5 | 19 | 21 |
| 10 | 10 | 28 | 29 |
| 15 | 15 | 21 | 17 |
| 20 | 20 | 40 | 32 |
| 25 | 25 | 42 | 45 |

**Extra**: Created another version with semaphore as flags and wait time to see if the results will be similar. Which we confirm as when N increases so does the wait time for each process.

|  |  |  |  |
| --- | --- | --- | --- |
| N | Dad | Son 1 | Son 2 |
| 5 | 5 | 10 | 10 |
| 10 | 16 | 13 | 23 |
| 15 | 26 | 30 | 53 |
| 20 | 45 | 32 | 42 |
| 25 | 65 | 56 | 35 |