School of computer Science

Progress Report of Simulation based assignment on

Multithreads program that implements Banker’s Algorithm

Submitted to: - Lovely Professional University

Faculty name: - Cherry Khosla

Submitted by: -

Name: - Md Amjad Ansari

Reg No: - 12114768

Roll No: - RK21SBB42

Section: - K21SB

Group: - 2

INTRODUCTION: - The banker’s algorithm is a resource allocation and deadlock avoidance algorithm that tests for safety by simulating the allocation for predetermined maximum possible amounts of all resources, then makes an “s-state” check to test for possible activities, before deciding whether allocation should be allowed to continue.

Banker’s algorithm is named so because it is used in banking system to check whether loan can be sanctioned to a person or not. Suppose there are n number of account holders in a bank and the total sum of their money is S. If a person applies for a loan then the bank first subtracts the loan amount from the total money that bank has and if the remaining amount is greater than S then only the loan is sanctioned. It is done because if all the account holders comes to withdraw their money then the bank can easily do it.
In other words, the bank would never allocate its money in such a way that it can no longer satisfy the needs of all its customers. The bank would try to be in safe state always.

ALGORITHM: -Banker's algorithm consists of two main algorithm

- Safety algorithm.
- Resource request algorithm

1. Safety algorithm: - The algorithm for finding out whether or not a system is in a safe state can be described as follows:

```
1) Let Work and Finish be vectors of length 'm' and 'n' respectively.
Initialize: Work = Available
Finish[i] = false; for i=1, 2, 3, 4….n
2) Find an i such that both
a) Finish[i] = false
b) $Need_i$ <= Work
if no such i exists goto step (4)
3) Work = Work + Allocation[i]
Finish[i] = true
goto step (2)
4) if Finish [i] = true for all i
then the system is in a safe state
```

2. Resource request algorithm: - Let Request$_i$ be the request array for process P$_i$. Request$_i$ [j] = k means process P$_i$ wants k instances of resource type R$_j$. When a request for resources is made by process P$_i$, the following actions are taken:

1) If Request$_i$ <= Need$_i$
Goto step (2) ; otherwise, raise an error condition, since the process has exceeded its maximum claim.
2) If Request$_i$ <= Available
Goto step (3); otherwise, P$_i$ must wait, since the resources are not available.
3) Have the system pretend to have allocated the requested resources to process Pi by modifying the state as
follows:
Available = Available – Requesti
Allocation$_i$ = Allocation$_i$ + Request$_i$
Need$_i$ = Need$_i$– Request$_i$

3. Implementation: -