**Operating system 2 Project – Cover sheet**

Project Title……Group#……

Discussion time :- …… Instructor ……

| ID | Name(Arabic) | Bounce | Minus | Total Grade | Comment |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

| Critrial | | Grade | Team Grade | Comment |
|---|---|---|---|---|
| Documentation | Solution pseudocode | 1 | | |
| | Examples of Deadlock | 1 | | |
| | How did solve deadlock | 1 | | |
| | Examples of starvation | 1 | | |
| | How did solve starvation | 1 | | |
| | Explanation for real world application and how did apply the problem | 1 | | |
| GitHub | Upload project files | 2 | | |
| | Submitted before discussion time (shared GitHub project link with TA and Dr) | 1 | | |
| | Only one contribution | -1 | | |
| Implementation | Run correctly (correct output) | 5 | | |
| | Run but with incorrect output | -3 | | |
| | Not run at all (error and exceptions) | -8 | | |
| | Free from Deadlock | 3 | | |
| | Free from deadlock in some cases and not free in other cases | -2 | | |
| | Free from Starvation | 2 | | |
| | Free from Starvation in some cases and not free in other cases | -1 | | |
| | Apply problem to real world application | 6 | | |
| Total | Total grade for Team | 25 | | |
| | Total Team Grade(after adjustment) | 25 | | |
| Bounce | Multithreading GUI Based Java Swing | +5 | | |
| | Multithreading GUI Based Java Swing(adjustment) | | | |
| | Multithreading GUI Based JavaFX | +10 | | |
| | Multithreading GUI Based JavaFX(adjustment) | | | |
| | Bounce Graphic and animation | +5 | | |
| Total with Bounce | Total Team Grade | | | |
| | Total Team Grade(after adjustment) | | | |

# *DINING PHILOSOPHIER'S PROBLEM DOCUMENTATION*

## SOLUTION PSEUDOCODE

```
do{
    wait (chopStick[i]);
        wait (chopStick [ (i+1) % size_of_phillosipher]);


        //Eat

        signal(chopStick[i]);
        signal(chopStick[ (i+1) % size_of_phillosipher]);

        //think
    } while (TRUE);
```

## EXAMPLE OF DEADLOCK

When all philosophers want to eat at the same time.

### HOW DID SOLVE THE DEADLOCK

Each philosopher will eat if each of the two neighbors don't want to eat or use a Chopstick.

## EXAMPLE OF STARVATION

When a philosopher wants to eat but can't eat because there are two neighbors who are eating for long time more than normal.

### HOW DID SOLVE THE STARVATION

For each philosopher we put a constant time + random time in fixed range.

## EXPLANATION FOR REAL WORLD APPLICATION

The problem of executing a transaction between two accounts is very similar.

To execute the transaction the thread must lock both accounts to ensure the correct value is debited from one account (assuming that there are always available funds) and crediting to another.

The topology is not exactly the round table but is very close. Imagine 5 accounts at the table. In this analogy, the accounts are chopsticks.
Any two accounts can participate in a transaction.
(Transactions == philosophers). So, in this example the transactions (philosopher) can not only sit at the edge of the table between two accounts (forks), but also on a line cutting across the table, connecting any two accounts (forks).