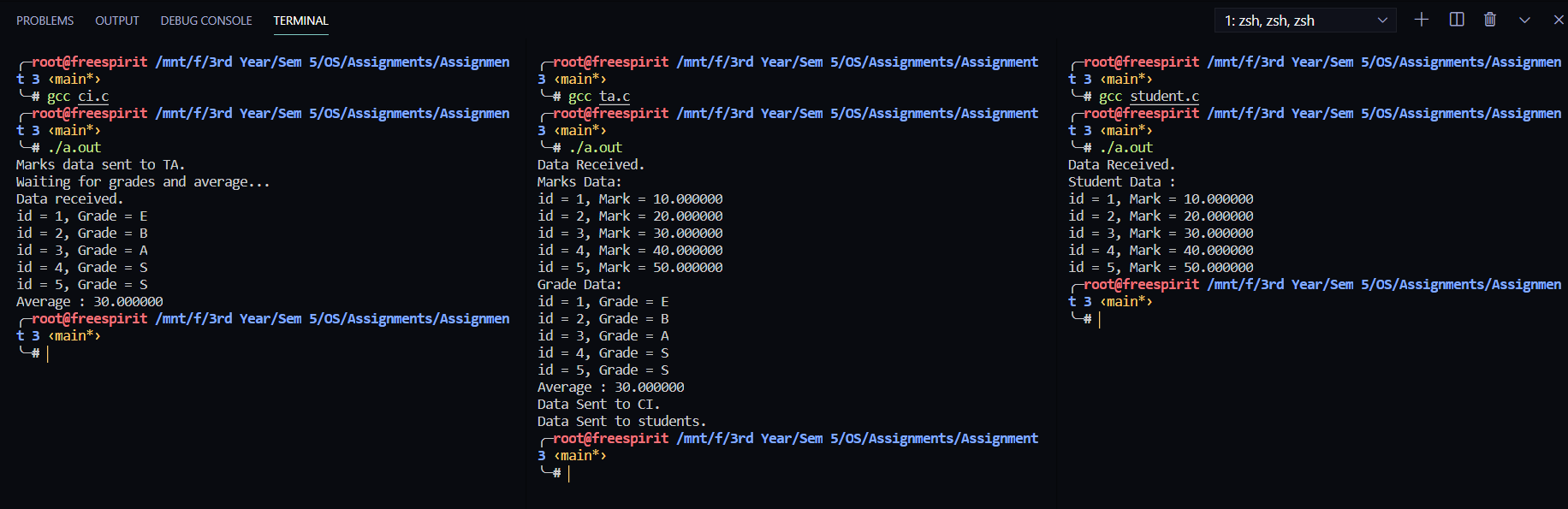
# OS Assignment 3

## Readme

*Meenakshi Madhu – B180390CS*

**Q1. Message Queue**

**Programs:**

* ci.c – Course Instructor
* ta.c – Teaching Assistant
* student.c – Student

**System calls used:**

* ftok(): to generate a unique key.
* msgget(): either returns the message queue identifier for newly created message queues or the identifier for an existing queue with the same key value.
* msgsnd(): Data is placed on to a message queue
* msgrcv(): messages are retrieved from a queue.
* msgctl(): It performs various operations on a queue. Here it is used to destroy message queue.

**Input array used:-**

inputmarks[5] = {10, 20, 30, 40, 50};

Here, the course instructor(CI) sends these marks to the Teaching assistant (TA) through the message queue, with a message-type 10. The TA receives these marks and calculates grades, and average marks. This new data is again written to the message queue with message-type as 100. The marks are written as well, with message-types 1-5 according to the student-id. The CI receives the grades, and average marks where as the student only receives their marks according to their student-id.

**Q2. Banker’s Algorithm**



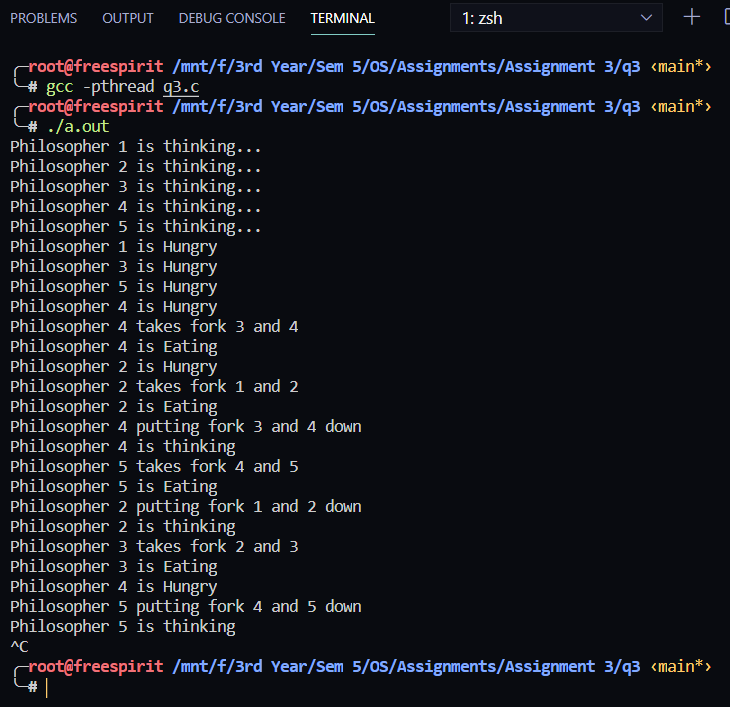
Input:

* No. of processes – n
* No. of resources – m
* Allocation table – AllocTable[n][m]
* Maximum Need table – MaxTable[n][m]
* Total no. of instances for m resources – Total[m]
* Sequence of processes

Using the available data, remaining resources needed for each process is calculated. According to this resource request for processes in the sequence, the program checks whether the request can be granted or not.

If for a particular process, the available resources are not enough, the given sequence is not a SAFE STATE. Else is it a SAFE STATE.

**Q3. Dining Philosopher’s problem using semaphores**



Here, the basic algorithm for an instance of the Dining Philosopher’s problem is implemented.

No. of philosophers = 5

An infinite loop is generated.