Implement a CPU scheduler on top of Linux using threads and compare scheduling algorithms

Task :

1. Generate N concurrent threads. Each can be Producer and Consumer. (Worker Threads)
2. Create the scheduler threads schedule the worker threads using Round Robin method.
3. Create the Reporter thread for printing the change of status of different worker threads.

Implementation :

1. Function used :
    a. `pthread_create` : To create the thread
    b. `pthread_kill` : To send the signal to specified thread
    c. `pthread_join` : To wait for the thread to complete
    d. `signal()` : For installing signal Handler
    e. `pthread_exit()` : Exit from thread
2. Shared Memory : The variables are declared globally so that they can be accessed by each thread.

Contribution :

1. Vedic Partap (16CS10053)
    a. Creating workers threads
    b. Create Scheduler thread
2. Rahul Kumar (16CS10042)
    a. Creating Reporter and Signal Handlers
    b. Create Report

How to Run :

```
$ g++ Ass4_3.cpp -lpthread -o thread
$./thread
```

Sample Output :

```
0: SUSPENDED -> RUNNING
0: RUNNING -> TERMINATED
2: SUSPENDED -> RUNNING
2: RUNNING -> SUSPENDED
3: SUSPENDED -> RUNNING
1: SUSPENDED -> RUNNING
1: RUNNING -> SUSPENDED
2: SUSPENDED -> RUNNING
2: RUNNING -> SUSPENDED
3: SUSPENDED -> RUNNING
3: RUNNING -> SUSPENDED
1: SUSPENDED -> RUNNING
1: RUNNING -> SUSPENDED
2: SUSPENDED -> RUNNING
2: RUNNING -> SUSPENDED
3: SUSPENDED -> RUNNING
3: RUNNING -> SUSPENDED
1: SUSPENDED -> RUNNING
1: RUNNING -> SUSPENDED
2: SUSPENDED -> RUNNING
2: RUNNING -> SUSPENDED
3: SUSPENDED -> RUNNING
3: RUNNING -> SUSPENDED
```