Name: Mohit Prajapati

Roll no.: AU1940171

Q.2(a)

Round Robin is Scheduling Algorithm, which is mostly used for multitasking. In Round-robin scheduling, each ready task runs turn by turn only in a cyclic queue for a limited time slice. This algorithm also offers starvation free execution of processes. Round robin is a pre-emptive algorithm. The CPU is shifted to the next process after fixed interval time, which is called time quantum/time slice. The process that is pre-empted is added to the end of the queue. Example of Round-Robin scheduling suppose the time slice=2. Burst time of the process P1, P2 and P3 are respectively 4,3 and5. If we apply Round Robbin in this example then we can get results like P1(0-2) P2(2-4) P3(4-6) P1(6-8) P2(8-9) P3(9-11) P3(11-12).

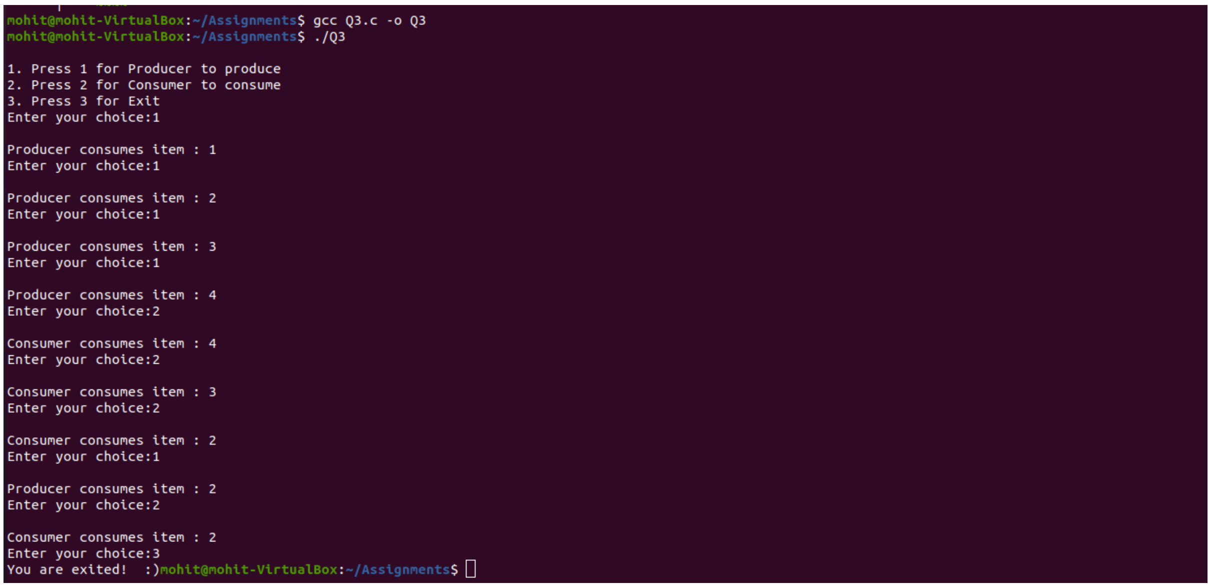
Q.2(b)

The buddy system is a memory allocation and deallocation algorithm that manages memory in power of 2 increments. Let's assume that size of the memory is 2^m and required size of memory is n. What this algorithm does is that if 2^(m-1) < n <= 2^m then allocate the whole block otherwise recursively divide the block equally and test the above condition each time, whenever it satisfies the condition, allocate the block and get out of the loop. Let's have an example suppose a system having buddy system with physical address space 256 KB. We have to calculate size of partition for 30 KB process. So 256→128,128 not satisfied and it will check again for 128→64,64 and again it is not satisfied 64→32,32 and so the answer is that size of partition for 30KB process=32 KB.

Q.3

First of all, we look at what semaphore do, Semaphores are integer variables that are used to solve the critical section problem by using two atomic operations, wait and signal that are used for process synchronization. The main aim of the semaphore is process synchronization and access control for a common resource in a concurrent environment. The same problem occurs in Producer and Consumer Problem. The producer consumer problem is a synchronization problem. Let's see what happens in this problem, Producer produces items and enter them into the fixed size of buffer. Consumer consumes it and remove from that buffer. What happens here is that producer and customer can not access the buffer at the same time. If consumer is removing from the buffer, then producer can't enter items into buffer at that time and same thing happen with producer and consumer. So here resource is only one and it is used by two persons. With the help of semaphores, we can apply process synchronization concept here. Let's assume that we have buffer of 4096-byte length. What producer thread does that they collect the data and writes into it and at the other hand consumer thread processes the collected data from the buffer. The objective here is that both the threads should not run at the same time. A mutex provides mutual exclusion, either producer or consumer can have the key and proceed with their work. As long as the buffer is filled by the consumer, the producer has to wait and it same apply for another case where the consumer has to wait. A semaphore is a generalized mutex. What we can do is that we can split the FOUR KB buffer into Four ONE KB buffers. What we can achieve from this is that the producer and consumer can work on different buffers at the same time.

In the code we have implemented this problem and basically idea is that we have to do certain operations like increase the buffer size, check the slots for the function of the producer, so that we can keep record of what is the size of buffer and all that. In the second function which is for the customer we have to decrease the size of the buffer and if the buffer short fall, then we have to also pop the message that buffer is empty. We have used 3 cases for these so when 1 is pressed producer function will work, when 2 is pressed consumer function will work and last number 3 is pressed you can exit from the code.



```
mohit@mohit-VirtualBox:~/Assignments$ gcc Q3.c -o Q3
mohit@mohit-VirtualBox:~/Assignments$ ./Q3

1. Press 1 for Producer to produce
2. Press 2 for Consumer to consume
3. Press 3 for Exit
Enter your choice:1

Producer consumes item : 1
Enter your choice:1

Producer consumes item : 2
Enter your choice:1

Producer consumes item : 3
Enter your choice:1

Producer consumes item : 4
Enter your choice:2

Consumer consumes item : 4
Enter your choice:2

Consumer consumes item : 3
Enter your choice:2

Consumer consumes item : 2
Enter your choice:1

Producer consumes item : 2
Enter your choice:2

Consumer consumes item : 2
Enter your choice:3
You are exited!  :)mohit@mohit-VirtualBox:~/Assignments$ 
```