سوال چهارم :

در این بخش 3 تا فایل shared_mem.h , consumer.c , producer.c داریم که به ترتیب حاوی کد تولید کننده ، مصرف کننده و تعاریف و اسامی استفاده شده در این برنامه ها هستند .

در producer :

در یک while بینهایت تولید کننده همواره در حال ایجاد یک عدد رندوم و قرار دادن ان در حافظه میباشد.

از ایندکس 0 حافظه مشترک ایجاد شده به عنوان پوینتر top استک ایجاد شده استفاده میشود، به این صورت که هر وقت یک عدد جدید تولید میشود، TOP یکی بالا میرود . یک سمافور اصلی داریم که دسترسی به حافظه مشترک را کنترل میکند و علاوه بر آن دو سمافور empty , full را داریم که هر وقت تولید کننده یک عدد میگذارد empty کم و full زیاد میشود به اندازه یکی . مقدار empty با 100 و full با 0 ،initialize شده .

در consumer :

در یک while بینهایت مصرف کننده همواره به دنبال فرصت برای برداشتن از حافظه مشترک است . به این صورت که مقدار Top را میخواند و اخرین عددی که تولید کننده قرار داده است را میخواند و TOP را یکی کم میکند و سپس empy را یکی افزایش و full را کاهش میدهد و sem را unlock میکند .

نکته مهمی که موجود است این است که producer بایستی قبل از مقدار دهی اولیه semaphore ها ، حافظه اشتراکی و سمافور های قبلی با این اسامی را unlink کنند و ببندد .

بخش الف :

در این بخش کنترلر نداریم و میتوانیم به یک تعداد دلخواه consumer داشته باشیم در کنار یک producer که در حال تولید عدد میباشد .

نمونه خروجی برنامه با در نظر گرفتن 2 مصرف کننده :

```
Top :95 Sent!: 20
Top :96 Sent!: 13
Top :97 Sent!: 7
Top :98 Sent!: 7
Top :99 Sent!: 15
Top :100 Sent!: 20
./consumer
./consumer
init sem
init sem
init map
init map
Top: 100  Recieved: 20
Top: 99  Recieved: 15
Top :99 Sent!: 16
Top :100 Sent!: 11
Top: 100  Recieved: 11
Top: 99  Recieved: 16
Top: 98  Recieved: 7
Top: 97  Recieved: 7
Top: 96  Recieved: 13
Top: 95  Recieved: 20
Top: 94  Recieved: 9
Top: 93  Recieved: 17
Top: 92  Recieved: 1
```

همانطور که دیده میشود پروسس های تولیدکننده و مصرف کننده ها به صورت موازی در حال ارسال و دریاف عدد هستند .

در بخش ب :

حال یک کنترلر داریم که دارای دسترسی read به حافظه اشتراکی هست و همواره میزان TOP را چک میکند . اگر این عدد از UP_LIMIT بالاتر باشد یک پروسس مصرف کننده جدید ایجاد میکند که TOP از این مقدار کمتر شود با سرعت بیشتر . و اگر از مقدار DOWN_LIMIT کمتر رفته باشد اخرین مصرف کننده جدید را kill میکند .

نمونه خروجی برنامه :

```
Top :94 Sent!: 9
Top :95 Sent!: 20
Top :96 Sent!: 13
Top :97 Sent!: 7
Top :98 Sent!: 7
Top :99 Sent!: 15
Top :100 Sent!: 20
./controller
init_controller
Top is upper than the UP_LIMIT..
init sem
init map
Top: 100  Recieved: 20
new process has been created with pid :6905 Top is upper than the UP_LIMIT..
Top :100 Sent!: 16
Top: 100  Recieved: 16
Top: 99  Recieved: 15
init sem
init map
Top: 98  Recieved: 7
Top: 97  Recieved: 7
Top: 96  Recieved: 13
Top: 95  Recieved: 20
```

```
Top: 93  Recieved: 5
Top: 92  Recieved: 20
Top: 91  Recieved: 4
Top: 90  Recieved: 7
Top :90 Sent!: 9
Top :91 Sent!: 9
Top :92 Sent!: 16
Top :93 Sent!: 1
Top :94 Sent!: 10
Top :95 Sent!: 17
Top :96 Sent!: 4
Top :97 Sent!: 19
Top :98 Sent!: 6
Top :99 Sent!: 7
Top :100 Sent!: 12
Top: 100  Recieved: 12
new process has been created with pid :6919 Top is upper than the UP_LIMIT..
Top: 99  Recieved: 7
Top: 98  Recieved: 6
init sem
init map
Top: 97  Recieved: 19
Top: 96  Recieved: 4
Top: 95  Recieved: 17
```