سوال یک :

در این سوال به ازای هر مشتری یک mutex ایجاد میشود که هرگاه عملیاتی بر روی حسابش انجام شود تنها توسط یک thread اتفاق بیفتد . همچنین یک mutex برای safebox در نظر گرفته شده زیرا تنها یک thread بایستی در لحظه از/به پول برداشت/واریز کند . برای ایجاد عملیات های هر مشتری به صورت رندم انجام میگیرد . یک amount رندوم و هم چنین customer id رندوم نیز برای هر transaction انتخاب میشود . به ازای هر transaction یک thread ایجاد میشود که یا باید پروسس برداشت یا پروسس واریز را انجام دهد . در پروسس برداشت، برای اجرا عملیات بر روی stock[customer_id] ابتدا mutex مربوط به این lock ، stock میشود سپس چک میشود که اگر با انجام این عملیات مقدار موجودی حساب منفی شد ، lock،safebox میشود و به میزان منفی از safebox برمیداریم . اگر safebox دارای مقدار کافی نبود منتظر میماند تا مقداری بزرگتر مساوی بدیهی این مشتری در safebox قرار گیرد تا بتواند عملیات برداشت خود را کامل کند . در عملیات واریز نیز ابتدا lock،mutex[customer_id] میشود سپس چک میشود که آیا مقدار stock ان منفی است یا خیر . اگر منفی بود به منزله بدیهی به safebox است پس به میزان min(amount , doubt_to_safebox) به safebox افزوده میشود و amount نیز از روی به stock مشتری اضافه میشود و در انتها mutex_unlock میشود. یک نمونه خروجی :

```
zahra@400522058:~/Desktop/git/IUST-OS4021/Assignments/HW4/Questions/Q1$ ./myprogram
Customer 3 Previous amount : 100.000000 stock was charged by 544.391330. New balance: 644.391330       safe-box value: 1000.000000
Customer 4 Previouse amount :100.000000 withdrew 320.572878 and 220.572878 from safe-box. New balance: -220.572878     safe-box value: 779.427122
Customer 5 Previouse amount :100.000000 withdrew 205.306495 and 105.306495 from safe-box. New balance: -105.306495     safe-box value: 674.120626
Customer 1 Previouse amount :100.000000 withdrew 404.981482 and 304.981482 from safe-box. New balance: -304.981482     safe-box value: 369.139145
Customer 5 Previous amount : -105.306495 stock was charged by 569.025863. New balance: 463.719367      safe-box value: 474.445640
Customer 5 Previous amount : 463.719367 stock was charged by 227.700692. New balance: 691.420059       safe-box value: 474.445640
zahra@400522058:~/Desktop/git/IUST-OS4021/Assignments/HW4/Questions/Q1$
```

سوال دوم :

در این سوال دو پروسس داریم که مسئول چاپ کردن حروف : F , E , G به ترتیب و A,C,B به ترتیب هستند و همچنین یک قانون داریم که A قبل از F و F قبل از Cبیاید . پس برای اینکه THREAD هایی که این پروسس هارا باید اجرا کنند موظف به رعایت این ترتیب شوند از دوتا SEMAPHORE استفاده میکنیم که باینری هستند و دارای مقدار اولیه 0 هستند.

```
    // INITIALIZING STEPS
    sem_init(&sem1, 0, 0);
    sem_init(&sem2, 0, 0);
    pthread_t th[2];
    pthread_create(&th[1], NULL, &p2, NULL);
    pthread_create(&th[0], NULL, &p1, NULL);

    pthread_join(th[0], NULL);
    pthread_join(th[1], NULL);
```

حال باید مطئین بشویم که اول A قبل از F چاپ میشود پس یک سمافور را مسئول قرار میدهیم که پس از چاپ A ، یک سیگنال POST بزند و اجازه بدهد F که منتظر چاپ A است نیز چاپ شود . برای رعایت ترتیب چاپ Fقبل از C نیز سمافور دیگر مسئول میکنیم و بعد از چاپ F سیگنال POSTمیزنیم که C از حالت WAIT دربیاید و چاپ شود توسط THREAD دوم که پروسس P2 را ران میکند

```
void* p1(void* arg)
{

    sem_wait(&sem1) ;
    printf("F\n") ;
    sem_post(&sem2) ;
```

```
    printf("E\n") ;
    printf("G\n") ;
    return arg;
}

void* p2(void* arg){
    printf("A\n") ;
    sem_post(&sem1);
    sem_wait(&sem2) ;
    printf("C\n") ;
    printf("B\n") ;
    return arg;
```

خروجی برنامه :

```
zahra@400522058:~/Desktop/git/IUST-OS4021/Assignments/HW4/Questions/Q2$ make
gcc -Wall -c -o Q2.o Q2.c
gcc -Wall -o myprogram Q2.o
zahra@400522058:~/Desktop/git/IUST-OS4021/Assignments/HW4/Questions/Q2$ ./myprogram
A
F
E
G
C
B
zahra@400522058:~/Desktop/git/IUST-OS4021/Assignments/HW4/Questions/Q2$
```