// Assignment 5 OS

Ans.1
// a) Binary Semaphore:-

```
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>
int a, b;
sem_t sem;
void ScanNumbers(void *ptr){
    for (;;){
        printf("%s", (char *)ptr);
        scanf("%d %d", &a, &b);
        sem_post(&sem);
        usleep(100 * 1000);
    }
}
void SumAndPrint(void *ptr){
    for (;;){
        sem_wait(&sem);
        printf("%s %d\n", (char *)ptr, a + b);
    }
}
int main()
{
    pthread_t thread1;
    pthread_t thread2;
    char *Msg1 = "Enter Number Two No\n";
    char *Msg2 = "sum = ";
    sem_init(&sem, 0, 0);
    pthread_create(&thread1, NULL, (void *)ScanNumbers, (void *)Msg1);
    pthread_create(&thread2, NULL, (void *)SumAndPrint, (void *)Msg2);
    pthread_join(thread1, NULL);
    pthread_join(thread2, NULL);
    printf("Wait For Both Thread Finished\n");
    sem_destroy(&sem);      `
    return 0;
}
```

// b) Counting Semaphore:-

```
#include <stdio.h>
#include <pthread.h>
#include <signal.h>
#include <semaphore.h>
#include <unistd.h>
sem_t s;
void handler(int signal)
{
    sem_post(&s);
```

```
}
void *singsong(void *param)
{
    sem_wait(&s);
    printf("I had to wait until your signal released me!\n");
}
int main()
{
    int ok = sem_init(&s, 0, 0);
    if (ok == -1) {
       perror("Could not create unnamed semaphore");
       return 1;
    }
    signal(SIGINT, handler);

    pthread_t tid;
    pthread_create(&tid, NULL, singsong, NULL);
    pthread_exit(NULL);
}
```

//Ans.2 Peterson's Algorithm

```
#include<pthread.h>
#include<stdio.h>
void *func1(void *);
void *func2(void *);
int flag[2];
int turn=0;
int global=100;
int main()
{
    pthread_t tid1,tid2;
    pthread_create(&tid1,NULL,func1,NULL);
    pthread_create(&tid2,NULL,func2,NULL);
    pthread_join(tid1,NULL);
    pthread_join(tid2,NULL);
}
void *func1(void *param)
{
    int i=0;
    while(i<2)
    {
        flag[0]=1;
        turn=1;
        while(flag[1]==1 && turn==1);
        global+=100;
        printf("FT: g: %d",global);
        flag[0]=0;
        i++;
    }
```

```
}
void *func2(void *param)
{
    int i=0;
    while(i<2)
    {
        flag[1]=1;
        turn=0;
        while(flag[0]==1 && turn==0);
        global-=75;
        printf("SP: g: %d",global);
        flag[1]=0;
        i++;
    }
}
```

// Ans.3 Race Condition using fork()

```
#include<unistd.h>
#include<stdlib.h>
int main()
{
    int pid,j;
    pid=fork();
    if(pid==0)
    {
        for(j=1;j<=10;j++)
         printf(1\nChild here\);
        exit(0);
    }
    else
    {
        for(j=1;j<=10;j++)
         printf(1\nParent here\);

    }
   exit(0);
}
```