```
/*  function to create deadlock */

void *function1();
void *function2();
pthread_mutex_t first_mutex, second_mutex;

int main() {
 pthread_mutex_init(&first_mutex,NULL);  //initialize the lock
 pthread_mutex_init(&second_mutex,NULL);
 pthread_t one, two;
 pthread_create(&one, NULL, function1, NULL);  // create thread
 pthread_create(&two, NULL, function2, NULL);
 pthread_join(one, NULL);
 pthread_join(two, NULL);
 printf("Thread joined\n");
 }

void *function1( ) {
    printf("Thread ONE trying to acquire first_mutex\n");
    pthread_mutex_lock(&first_mutex);  // to acquire the resource/mutex lock
    printf("Thread ONE acquired first_mutex\n");
    sleep(1);

    printf("Thread ONE Trying to acquire second_mutex\n");
    pthread_mutex_lock(&second_mutex);
    printf("Thread ONE acquired second_mutex\n");
    pthread_mutex_unlock(&first_mutex); // to release the resource
    printf("Thread ONE released first_mutex\n");



}
```

```
void *function2( ) {
    printf("Thread TWO trying to acquire second_mutex\n");
    pthread_mutex_lock(&second_mutex);
    printf("Thread TWO acquired second_mutex\n");
    sleep(1);
    printf("Thread TWO trying to acquire first_mutex\n");
    pthread_mutex_lock(&first_mutex);
    printf("Thread TWO acquired first_mutex\n");
    pthread_mutex_unlock(&second_mutex);
    printf("Thread TWO released second_mutex\n");
}



/* function to prevent deadlock */



for(i=0;i<n;i++)
                        {
                                    p[i]=i+1;
                                    bt[i]=1;
                                    printf("\t Who arrived %d? =",i+1);
                                    scanf("%d",&pr[i]);
                                    if(pr[i]==0)
                                    {
                                                c=c+1;
                                                //printf("%d",c);
                                    }
                                    if(pr[i]==0 && d==1)
                                    {
```

```
                //printf("inside");
                pr[i]=pr[i]+1;//printf("\t%d",pr[i]);
                d=0;
        }
        if(c==3)
        {
                c=0;
                d=1;
                continue;
        }
}
for(i=0;i<n;i++)
{
        max=i;
        for(j=i+1;j<n;j++)
        {
                if(pr[j] <pr[max])
                max=j;
        }
        temp=pr[max];
        pr[max]=pr[i];
        pr[i]=temp;
        temp=bt[max];
        bt[max]=bt[i];
        bt[i]=temp;
        temp=p[max];
        p[max]=p[i];
        p[i]=temp;
}

for(i=0;i<n;i++)
```

```
          {
                    wt[i+1]=bt[i]+wt[i];
                    ta[i]=bt[i]+wt[i];
                    sum+=ta[i];
          }
          for(i=0;i<n;i++)
          {
                    printf("\nWaiting time for person %d is =%d",p[i],wt[i]);
                    //printf("\t turn around time for p[%d]=%d",p[i],ta[i]);
          }
          //printf("\n\naverage turn around=%d",sum/n);
          printf("\n\nScene 3 at Library is complete\n");
          int a;
          printf("\nHit 0 to enter other scene, else any:  ");
          scanf("%d",&a);
          if (a==0)
          {
                    main();
          }
          else {exit1();}
          break;
}
```