1. Να χρησιμοποιηθούν κατάλληλα οι συναρτήσεις shmget/shmat/shmctl με όλα τα ορίσματα. Το μέγεθος της μνήμης να πιάνει χώρο 4 byte.
   int shmid;
   int *myvars;

2. Δύο ψαράδες πηγαίνουν για ψάρεμα με μια βάρκα, έστω ότι αναπαρίστανται από τις συναρτήσεις fishermanA και fishermanB. Καθένας από τους δύο κάνει τα εξής: περπατά μέχρι την αποβάθρα (συνάρτηση walk_to_dock()), αν ο άλλος δεν είναι εκεί τον περιμένει, μπαίνει στη βάρκα (board()) και φεύγουν μαζί. Να συμπληρώσετε κώδικα (αν και όπου χρειάζεται) με χρήση semaphores έτσι ώστε οι ψαράδες να μπαίνουν στην βάρκα μόνο όταν και οι δύο έχουν φτάσει στην αποβάθρα. Να γίνει κατάλληλη αρχικοποίηση των semaphores. Δίδεται σκελετός:

   sem_t semA, semB;
   //Αρχικοποίηση

   ```
   void fishermanA(){
           walk_to_dock();


           board();

   }
   ```

   ```
   void fishermanB(){
           walk_to_dock();


           board();

   }
   ```

3. Θεωρήστε σύστημα τραπεζικών συναλλαγών όπου πραγματοποιείται μεγάλος αριθμός πράξεων ταυτόχρονα. Δίδεται συνάρτηση για την υλοποίηση ατομικής μεταφοράς ενός ποσού από έναν λογαριασμό σε έναν άλλο. Μπορεί η χρήση της συνάρτησης να οδηγήσει σε αδιέξοδο; Αν όχι δικαιολογήστε το σκεπτικό σας, αν ναι δώστε παράδειγμα και λύση ώστε να αποφύγετε το πρόβλημα αδιέξοδου.

   ```
   struct account{
       int balance;
       pthread_mutex_t L = PTHREAD_MUTEX_INITIALIZER;
   };

   money_xfer(struct account *src, struct account *dst, int amount)
   {
   Pthread_mutex_lock(&src->L) // lock A account
   Pthread_mutex_lock(&dst->L) // lock B account
       src->balance-=amount; // remove amount from src account
       dst->balance+=amount; //  and add to dst account
   pthread_mutex_unlock(&dst->L) // unlock B
   pthread_mutex_unlock(&src->L) // unlock A
   }
   ```

ΛΥΣΕΙΣ

1. Να χρησιμοποιηθούν κατάλληλα οι συναρτήσεις shmget/shmat/shmctl με όλα τα ορίσματα. Το μέγεθος της μνήμης να πιάνει χώρο για 4 byte.
   int shmid;
   int *myvars;

```
shmid=shmget(IPC_PRIVATE,4, IPC_CREAT|0666);
   if(shmid < 0)
      printf("ERROR\n");
   else
      printf("Shmid Is %d\n",shmid);
myvars=shmat(shmid,0,0);
shmctl(shmid,IPC_RMID,0);
```

2. Δύο ψαράδες πηγαίνουν για ψάρεμα με μια βάρκα, έστω ότι αναπαρίστανται από τις συναρτήσεις fishermanA και fishermanB. Καθένας από τους δύο κάνει τα εξής: περπατά μέχρι την αποβάθρα (συνάρτηση walk_to_dock()), αν ο άλλος δεν είναι εκεί τον περιμένει, μπαίνει στη βάρκα (board()) και φεύγουν μαζί. Να συμπληρώσετε κώδικα (αν και όπου χρειάζεται) με χρήση semaphores έτσι ώστε οι ψαράδες να μπαίνουν στην βάρκα μόνο όταν και οι δύο έχουν φτάσει στην αποβάθρα. Να γίνει κατάλληλη αρχικοποίηση των semaphores.
   Δίδεται σκελετός:

```
sem_t semA, semB;
//Arxikopoihsh
…. sem_init(&semA, 0, 0);
…. sem_init(&semB, 0, 0);
```

```
void fishermanA(){
        walk_to_dock();
        sem_post(&semB);
        sem_wait(&semA);
        board();

}
```

```
void fishermanB(){
        walk_to_dock();
        sem_post(&semA);
        sem_wait(&semB);
        board();

}
```

3. Θεωρήστε σύστημα τραπεζικών συναλλαγών όπου πραγματοποιείται μεγάλος αριθμός πράξεων ταυτόχρονα. Δίδεται συνάρτηση money_xfer για την υλοποίηση ατομικής μεταφοράς ενός ποσού από έναν λογαριασμό σε έναν άλλο. Α) Μπορεί η χρήση της συνάρτησης να οδηγήσει σε αδιέξοδο; Αν όχι δικαιολογήστε το σκεπτικό σας, αν ναι δώστε Β) λύση ώστε να αποφύγετε το πρόβλημα αδιέξοδου.

```
struct account{
    int balance;
    pthread_mutex_t L = PTHREAD_MUTEX_INITIALIZER;
};

money_xfer(struct account *src, struct account *dst, int amount)
{
Pthread_mutex_lock(&src->L) // lock A account
Pthread_mutex_lock(&dst->L) // lock B account
    src->balance-=amount; // remove amount from src account
    dst->balance+=amount; //  and add to dst account
pthread_mutex_unlock(&dst->L) // unlock B
pthread_mutex_unlock(&src->L) // unlock A
}
```

Απάντηση:

Η συνάρτηση μπορεί να οδηγήσει σε deadlock. Αν έχουμε δύο λογαριασμούς Alice και Bob, τότε ταυτόχρονη κλήση των money_xfer(Alice, Bob) και money_xfer(Bob, Alice) μπορεί να οδηγήσει σε κυκλική αναμονή και deadlock.

Επίσης σχεδιάστε παρόμοια περίπτωση deadlock με 3 λογαριασμούς (Alice, Bob, Charlie).

```
pthread_mutex_lock(&src) ); // lock src account
If (pthread_mutex_trylock(&dst) ){ // try lock dst account
    pthread_mutex_unlock(&src)
}else{
    src->balance-=amount; // remove amount from src account
    dst->balance+=amount; //  and add to dst account
    pthread_mutex_unlock(&dst->L) // unlock B
    pthread_mutex_unlock(&src->L) // unlock A
    }

}
```