Problema 1

```
#include <pthread.h>
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>

#define MAX_RESOURCES 5
int available_resources = MAX_RESOURCES;

pthread_mutex_t mtx;

int decrease_count(int count)
{
   while (1)
   {
      pthread_mutex_lock (&mtx);

      if (available_resources < count)
         pthread_mutex_unlock (&mtx);
      else
      {
         available_resources -= count;
         printf("Got %d resources, remaining %d resources\n", count,
available_resources);
         pthread_mutex_unlock (&mtx);
         return 0;
      }
   }
}

int increase_count(int count)
{
   pthread_mutex_lock (&mtx);
   available_resources += count;
   printf("Release %d resources, remaining %d resources\n", count,
available_resources);
   pthread_mutex_unlock (&mtx);
   return 0;
}

void *f(void *arg)
{
   int* count = (int*)arg;
   decrease_count(*count);
   increase_count(*count);
   return NULL;
}

int main()
{
   pthread_mutex_lock (&mtx);
   if (pthread_mutex_init (&mtx, NULL)){
      perror(NULL);
      return errno;
   }
   printf("MAX RESOURCES = %d\n", available_resources);

   pthread_t t[5];
   for (int i=0; i<5; i++)
   {
      int *count = (int *) malloc(sizeof(int));
      *count = rand()%(MAX_RESOURCES + 1);
      if (pthread_create(&t[i], NULL, f, count))
      {
         perror(NULL);
         return errno;
      }
   }

   for (int i=0; i<5; i++)
   {
      if (pthread_join(t[i], NULL))
      {
         perror(NULL);
         return errno;
      }
   }

   pthread_mutex_destroy(&mtx);

   return 0;
}
```

Problema 2

```
#include <pthread.h>
#include <stdio.h>
#include <errno.h>
#include <stdlib.h>
#include <semaphore.h>

#define NTHRS 5

pthread_mutex_t mtx;
sem_t sem;
int global_s = 0;

int barrier_pt()
{
   int local_s;
   pthread_mutex_lock (&mtx);
   global_s ++;
   local_s = global_s;
   pthread_mutex_unlock (&mtx);

   if (local_s < NTHRS)
      if (sem_wait(&sem))
      {
         perror(NULL);
         return errno;
      }
      else ;
   else
      for (int i=0; i<NTHRS; i++)
         sem_post(&sem);
}

void * tfun(void *v)
{
   int *tid = (int *)v;
   printf ("%d reached the barrier\n", *tid);
   barrier_pt ();
   printf ("%d passed the barrier\n", *tid);
   free(tid);
   return NULL;
}

int main()
{
   pthread_t t[10];

   if (sem_init(&sem, 0, global_s))
   {
      perror(NULL);
      return errno;
   }

   if (pthread_mutex_init(&mtx, NULL))
   {
      perror(NULL);
      return errno;
   }

   for(int i=0; i<NTHRS; i++)
   {
      int *a = (int*) malloc(1);
      *a = i;
      if (pthread_create(&t[i], NULL, tfun, a))
      {
         perror(NULL);
         return errno;
      }
   }

   for(int i=0; i<NTHRS; i++)
      if (pthread_join(t[i], NULL))
      {
         perror(NULL);
         return errno;
      }

   sem_destroy(&sem);
   pthread_mutex_destroy(&mtx);

   return 0;
}
```