```c
#include <stdio.h>
#include <pthread.h>
#include <windows.h>

pthread_t t1;
pthread_t t2;
pthread_t t3;  //nomeando as threads
pthread_mutex_t lock;

void* tarefa1(void* rec){ //funcao para que será exercida pela primeira thread, somando apenas um número
  int* recurso = (int*) rec;
  int c;
  for (c = 0; c < 10; c++){
    pthread_mutex_lock(&lock); //trancando o acesso ao recurso, para que não haja concorrência
    *recurso = *recurso + 1;
    printf("Thread 1, ao disputar o recurso com outras threads, conseguiu somar com 1, recurso = %d\n\n", *recurso);
    pthread_mutex_unlock(&lock); //destrancando o acesso ao recurso
    Sleep(1000);
  }
  return NULL;
}

void* tarefa2(void* rec){ //funcao para que será exercida pela segunda thread, somando dobrando o recurso
  int* recurso = (int*) rec;
  int c;
  for (c = 0; c < 10; c++){
    pthread_mutex_lock(&lock);//trancando o acesso ao recurso, para que não haja concorrênci
    *recurso = *recurso * 2;
    printf("Thread 2, ao disputar o recurso com outras threads, conseguiu dobra-lo, recurso = %d\n\n", *recurso);
    pthread_mutex_unlock(&lock);  //destrancando o acesso ao recurso
    Sleep(2000);
  }
  return NULL;
}

void* tarefa3(void* rec){
  int* recurso = (int*) rec;
  int c;
  for (c = 0; c < 10; c++){
    pthread_mutex_lock(&lock);//trancando o acesso ao recurso, para que não haja concorrênci
    *recurso = *recurso - 1;
    printf("Thread 3, ao disputar o recurso com outras threads, conseguiu subtrair 1, recurso = %d\n\n", *recurso);
    pthread_mutex_unlock(&lock); //destrancando o acesso ao recurso
```

```
    Sleep(5000);
  }
  return NULL;
}

int main(void){
  int recurso = 1;
  pthread_mutex_init(&lock, NULL); //criando o mutex
  printf("Inicio\n");

  pthread_create(&t1, NULL, tarefa1, &recurso);
  pthread_create(&t2, NULL, tarefa2, &recurso);
  pthread_create(&t3, NULL, tarefa3, &recurso);// vinculando as threads as funções

  pthread_join(t1, NULL);
  pthread_join(t2, NULL);
  pthread_join(t3, NULL); // iniciando as threads com suas funcoes

  pthread_mutex_destroy(&lock);//finalizando o mutex
  printf("Fim\n");
  return 0;
}
```