Abgabe 1

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

void* function(void* arg) {
  //sleeps the given time
  sleep((int) arg);
  int id = pthread_self();
  printf("This is thread %d\n", id);
  //returns its id
  return (void *) id;
}

int main(int argc, char *argv[]) {
  pthread_t thread_one;
  pthread_t thread_two;
  pthread_attr_t attr;
  pthread_attr_t attr1;
  pthread_t returnValue;
  pthread_t returnValue1;
  int err;

  /*
   * set attributes
   * PTHREAD_CREATE_JOINABLE so that the process won't be closed until it
    is joined.
   * If PTHREAD_CREATE_DETACHED is used, the main program wouldn't be able
    to join the thread.
   * With detached as soon as the thread is finished it gets deleted.
   * With joinable when the thread is finished it gets preserved until it
    gets joined.
   */
  pthread_attr_init(&attr);
  pthread_attr_init(&attr1);
  pthread_attr_setdetachstate(&attr, PTHREAD_CREATE_JOINABLE);
  pthread_attr_setdetachstate(&attr1, PTHREAD_CREATE_JOINABLE);

  //create first thread that sleeps 2 seconds
  err = pthread_create(&thread_one, &attr, &function, (void *) 2);

  if (err != 0) {
    printf("pthread_create: %d ", strerror(err));
  }

  //create second thread that sleeps 4 seconds
  err = pthread_create(&thread_two, &attr1, &function, (void *) 4);
  if (err != 0) {
    printf("pthread_create: %d  ", strerror(err));
  }

  printf("Joining Thread One\n");
  //join first thread and get returned thread id
  err = pthread_join(thread_one, (void **) &returnValue);
  if (err != 0) {
    printf("pthread_join: %d ", strerror(err));
  }
  printf("returnValue %d\n", returnValue);

  printf("Joining Thread Two\n");
  //join second thread and get returned thread id
  err = pthread_join(thread_two, (void **) &returnValue1);
  if (err != 0) {
    printf("pthread_join: %d ", strerror(err));
  }
  printf("returnValue %d\n", returnValue1);

  //compare returned thread id with the one received while creating
  printf("Difference for first thread %d\n", returnValue - thread_one);
  printf("Difference for second thread %d\n", returnValue1 - thread_two);

  return EXIT_SUCCESS;
}
```