CSE 430 - Operating Systems
Fall 2015
Project #2

Due Date: Oct 14th 2015
(groups of 2, use same group as Proj 1, unless there is a problem)

Overview:
Using the queuing routines (not all will be used) this project will implement the ability to run multiple functions as threads, using non-preemptive scheduling.

Step 1: TCB and context:
The queue items defined in your q.h file has to changed, to be of type TCB_t. The TCB_t and an initialization routine are provided in a header file tcb.h. Note that TCB_t has previous and next pointers along with an ucontext_t field to store a thread context.

There is a routine in the tcb.h file called init_TCB, which is used to initialize a TCB for a new thread. The arguments to init_TCB are:

```
1. pointer to the function, to be executed
2. pointer to the thread stack
3. size of the stack
```

Step 1 consists of understanding tcb.h and changing the q-element type in your q.h file to TCB_t

Step 2:
Write a routine called start_thread and put it in a file called "**threads.h".** Also routines defined in Step 3 will go into this file. You will need to include q.h into threads.h. The code for start_thread looks like:

```
void start_thread(void (*function)(void))
 { // begin pseudo code
      allocate a stack (via malloc) of a certain size (choose 8192)
      allocate a TCB (via malloc)
      call init_TCB with appropriate arguments
      call addQ to add this TCB into the "RunQ" which is a global header pointer
   //end pseudo code
 }
```

Step 3:
Write the routines called "yield" and "run" which cranks up the works (and put them in threads.h). These routines are defined as:

```
void run()
{   // real code
    ucontext_t parent;     // get a place to store the main context, for faking
    getcontext(&parent);   // magic sauce
    swapcontext(&parent, &(RunQ->conext));  // start the first thread
 }

void yield() // similar to run
 { // begin pseudo code
   rotate the run Q;
   swap the context, from previous thread to the thread pointed to by RunQ
  // end pseudo code
 }
```

Step 4:
Write a driver program in a file called thread_test.c. Into thread_test.c.c include threads.h which includes q.h, which includes TCB.h, which includes ucontext.h.

Declare a global RunQ.

Write a few functions with infinite loops (put a yield in each loop). Note: try to write thread functions that are

meaningful, use global and local variables

In main, initialize RunQ, start threads using the functions you defined using start_thread.

Call run() and watch.

Call friends and family

SUBMIT:

Your project must consist of 4 files

1. TCB.h (uses ucontext.h) // this can be a copy of the file provided
2. q.h (includes TCB.h)
3. threads.h (includes q.h)
4. thread_test.c (includes threads.h) – must contain your name(s) in comments @ beginning
   (make sure the compile command, "gcc thread_test.c" does the correct compilation).

All 4 files are to be ZIPPED into one zip or gzip file. The name of this file should reflect the name(s) of the people submitting (abbreviated, do not make it too long).

Email the zip file to 430<dot>proj at gmail.

Note: Grading is on Ubuntu. It is in your interest to make sure the program compiles and runs in the target test platform.