CSSE 332 – Operating Systems
Rose-Hulman Institute of Technology
Computer Science and Software Engineering Department

Exam 1 — Computer Part

Name:_____ Section:_____ CM:_____

Instructions: This part of the exam is open notes (same as paper part) and open computer.

- You must disable all chat tools (IM, ICQ, IRC, Lync, etc.) before the exam starts.
- **Any communication with anyone other than an exam proctor during the exam could result in a failing grade for the course**.
- You may refer to programs you have written for the course-both those assigned and those you have written for practice. You may not, however, refer to programs written by others except those provided in your textbook or by the course staff.
- Regarding materials on the web, you may only use the course web site and pages linked directly from it. Of course, search engine use is not allowed.
- Write all answers on these pages.
- Submit all code and support files to your svn repository.
- Read the entire examination before starting, then budget your time.

Problem	Points available	Your points
1	14	
2	10	
3	13	
4	13	
5	10	
Total	60	

Checkout the `ThreadProblemSolver` project from your SVN repository into your local workspace. All of your work should be done in this project.

`ThreadProblemSolver` is a simple program that lets you spawn (and eventually cancel) threads that solve quadratic equations and display their roots. The program is currently able to create threads to solve a quadratic equation with equal roots.

The `ThreadProblemSolver` project contains 4 ".c" files, and 4 ".h" files. The code cannot currently be built using `make` (a `Makefile` has not been completed yet), but it can be compiled with the gcc compilation command included in the comments at the top of the file `threadProblemSolver.c`.

Here's a transcript of running `ThreadProblemSolver`:

```
ThreadProblemSolver> ./threadProblemSolver
Enter 'r' to create a new thread that computes real distinct roots
for a quadratic equation.
Enter 'e' to create a new thread that computes roots for a quadratic
equation with equal roots.
Enter 'c' to create a new thread that computes complex roots for a
quadratic equation.
Enter 'k' to cancel the most recently started thread.
Enter 'q' to terminate the application.
e
Create EQUAL!
Hello from thread EQUAL!
================================================================
Quadratic equation = 1x^2+2x+1 = 0
Has equal roots.
Root of quadratic equation is: -1.000
================================================================
r
This command is not supported.
c
This command is not supported.
q
```

Notice that real and complex roots have not been implemented yet.

For each thread created the master thread should print "Create THREAD_NAME!" where THREAD_NAME is replaced by EQUAL, REAL or COMPLEX depending on the type of the quadratic equation's roots. Each thread should print its "Hello" message based on the type of roots, compute the roots of its quadratic equation, and then sleep for two second. Instead of exiting at this point, the thread should remain in a loop ready to recompute roots as long as the variable `shouldRun` is true.

The master thread uses a linked list to store thread information. The most recently created thread is accessible via the variable `latestThread`. See `createThread()` for details on how the linked list is built.

Recall from math that a quadratic equation is typically written in the form $ax^2 + bx + c = 0$. The roots are computed using the formular

$$r = \frac{-b \pm \sqrt{d}}{2a}$$

where

$$d = b^2 - 4ac.$$

d can also be used to determine type of the roots:

$d < 0$ – the equation has complex roots
$d = 0$ – the equation has real roots that are the same
$d > 0$ – the equation has distinct real roots.

Use this information to solve the problems posed in this exam.

Be sure to commit your work often so you do not miss committing it at the end of the exam.

Problem 1 (14 points) Modify `ThreadProblemSolver` so that it supports creating threads to solve a quadratic equation with distinct real roots.

You should add your thread information to the linked list described above. The results of running your solution should resemble the following transcript.

```
...

e
Create EQUAL!
Hello from thread EQUAL!
================================================
Quadratic equation = 1x^2+2x+1 = 0
Has equal roots.
Root of quadratic equation is: -1.000
================================================
r
Create REAL!
Hello from thread REAL!
================================================
Quadratic equation = 1x^2-10x-75 = 0
Has real number roots.
Roots of quadratic equation are -5.000 and 15.000
================================================
```

Problem 2 (10 points) Modify `ThreadProblemSolver` so that it supports creating threads to solve a quadratic equation with complex roots.

The results of running your solution should resemble the following transcript.

```
...

e
Create EQUAL!
Hello from thread EQUAL!
================================================================
Quadratic equation = 1x^2+2x+1 = 0
Has equal roots.
Root of quadratic equation is: -1.000
================================================================
r
Create REAL!
Hello from thread REAL!
================================================================
Quadratic equation = 1x^2-10x-75 = 0
Has real number roots.
Roots of quadratic equation are -5.000 and 15.000
================================================================
c
Create COMPLEX!
Hello from thread COMPLEX!
================================================================
Quadratic equation = 1x^2-10x+75 = 0
Has complex roots.
Roots of quadratic equation are:
5.000+7.071i
5.000-7.071i
================================================================
```

Problem 3 (13 points) Add a "k" command to `ThreadProblemSolver`. Typing "k" should cancel the most recent thread. Typing "k" again should cancel the second most recent thread, etc. In essence, a stack of threads exists and "k" cancels the thread at the top the stack. When no threads are running, typing "k" should display the message: "No threads running!"

Once a thread is canceled, the master thread should print "thread THREAD_NAME stopped" after successfully executing join.

To cancel a thread safely, you will need to use the `shouldRun` flag in the thread data structure. When the thread needs be canceled, have your master thread set the value to 0. That should terminate the loop in the `runner()` function that keeps the thread running. You solution should yield results that resemble the transcript below.

```
Enter 'r' to create a new thread that computes real distinct
roots for a quadratic equation.
Enter 'e' to create a new thread that computes roots for a
quadratic equation with equal roots.
Enter 'c' to create a new thread that computes complex roots
for a quadratic equation.
Enter 'k' to cancel the most recently started thread.
Enter 'q' to terminate the application.
r
Create REAL!
Hello from thread REAL!
================================================================
Quadratic equation = 1x^2-10x-75 = 0
Has real number roots.
Roots of quadratic equation are -5.000 and 15.000
================================================================
e
Create EQUAL!
Hello from thread EQUAL!
================================================================
Quadratic equation = 1x^2+2x+1 = 0
Has equal roots.
Root of quadratic equation is: -1.000
================================================================
c
Create COMPLEX!
Hello from thread COMPLEX!
================================================================
Quadratic equation = 1x^2-10x+75 = 0
Has complex roots.
Roots of quadratic equation are:
5.000+7.071i
5.000-7.071i
================================================================
k
Thread COMPLEX stopped.
k
Thread EQUAL stopped.
k
Thread REAL stopped.
k
No threads running!
q
```

Problem 4 (13 points) Add an "a" command to `ThreadProblemSolver`. Typing "a" should verify that threads which have not been canceled are still alive. A thread verifies its "aliveness" by redoing its computations just as when it was created.

An easy way to do this is by resetting the `type` field in the thread specific data structure of each thread to what it was when the thread was created. Doing so should yield results that resemble the listing below.

```
Enter 'r' to create a new thread that computes real distinct
roots for a quadratic equation.
Enter 'e' to create a new thread that computes roots for a
quadratic equation with equal roots.
Enter 'c' to create a new thread that computes complex roots
for a quadratic equation.
Enter 'k' to cancel the most recently started thread.
Enter 'a' to allow threads that remain to redo computations.
Enter 'q' to terminate the application.
a
No threads available!
r
Create REAL!
Hello from thread REAL!
================================================================
Quadratic equation = 1x^2-10x-75 = 0
Has real number roots.
Roots of quadratic equation are -5.000 and 15.000
================================================================
a
================================================================
Quadratic equation = 1x^2-10x-75 = 0
Has real number roots.
Roots of quadratic equation are -5.000 and 15.000
================================================================
e
Create EQUAL!
Hello from thread EQUAL!
================================================================
Quadratic equation = 1x^2+2x+1 = 0
Has equal roots.
Root of quadratic equation is: -1.000
================================================================
a
================================================================
Quadratic equation = 1x^2+2x+1 = 0
Has equal roots.
Root of quadratic equation is: -1.000
```

```
================================================================
================================================================
Quadratic equation = 1x^2-10x-75 = 0
Has real number roots.
Roots of quadratic equation are -5.000 and 15.000
================================================================
q
```

Problem 5 (10 points) Complete the `Makefile` for `threadProblemSolver`.

(a) Use the gcc compilation command included in the comments at the top of the file `threadProblemSolver.c` to create a `Makefile` that compiles the entire code base and produces the `threadProblemSolver` executable.

(b) Your `Makefile` might recompile the entire code base every time there is a change to any file. Modify your `Makefile` to generate ".o" files and then use the ".o" files to compile the final executable. When a specific source file (".h" or ".c") is updated, your `Makefile` should only rebuild what is necessary to produce a new version of the executable, reusing the ".o" files for parts of the system that have not changed. Recall that the "–c" directive will make `gcc` compile without linking (i.e. produce ".o" files).

Have you committed your files and changes to svn?