Threads Homework

Objectives
- Utilize the C programming language.
- Understand threads.
- Understand the shared memory differences between processes and threads.
- Practice good programming techniques.

Program Requirements
- You <u>MUST</u> use the C programming language for this assignment.
- Your program filename must be named threads_YourFullName.c (threads_JoeStudent.c).
- You must follow the department documentation standards.
- The program must follow good general programming practices. For example, utilize functions, error handling, free memory, etc. For error handling, be sure to utilize perror() and/or strerror() appropriately and include a user-friendly error message.
- The program must follow recommend programming for creating threads, which means ending all threads and the process appropriately.
- There is no need to worry about mutual exclusion (locks) for this program.

You will write a program that utilizes threads. The program must meet the following specifications:
- Declare a global integer variable, *x*. Yes, a global variable. No, this is not recommended in most circumstances.

The function *main* must do the following:
- Accept one command-line argument (CLA) that will specify the number of threads to create. Be sure to do valid checking with a usage clause. Assign this value to *numThreads*. The number of threads must be between 1 and 10, inclusive (not less than 1 and not more than 10).

- Assign 10 to *x*.
- Declare an integer variable, *num*, and assign it an initial value of 50.
- Declare an integer pointer variable, *nump*, and assign it an initial value of 100.

- Print a statement indicating the program is creating threads, the value of *numThreads*, the current value of *num,* the current value of *nump*, and the current value of *x.*
- Create *numThreads* concurrent threads.

The <u>master thread</u> will:
- Do no processing related to the variables *num*, *nump* and *x*.
- It will simply coordinate the sub-threads.

Each sub-thread will do the following:

- Print the following information: “BEFORE increment”, thread number, thread ID, value of *num*, value of *nump*, and value of *x*.
- Add 5 to the value of *x*.
- Increment the value of *num*.
- Increment the value of *nump*.
- Print the following information: “AFTER increment”, thread number, thread ID, value of *num*, value of *nump*, and value of *x*.
- End thread

The print statements for the sub-thread should follow this format (note the values may be different):

```
BEFORE increment: Thread number 1 with TID of 3795580672, num = 100, nump = 500, x = 5
AFTER increment: Thread number 1 with TID of 3795580672, num = 101, nump = 500, x = 5
```

Submission

- Submit your program via turnin on acad.