Homework 2

- There are 4 +1 (optional) problems
  - (4 points) Problem 0 - Optional
    - Have a programming environment
    - Virtual Machine and Linux or Unix related OS
    - Do coding in that environment.
  - (1 points) Problem 1
    - Modify the given code to have each thread counts.
  - (2 points) Problem 2
    - Modify the code in Problem 2 to print numbers in increasing order.
  - (3 points) Problem 3
    - Implement `calsum_multithread()` and compare with `calsum()`
  - (5 points) Problem 4
    - Implement the bounded buffer problem

Problem 0 (optional)

You will get additional point if you

- Install virtual machine on your computer.
  you can use any virtual machine (ex: VMware, VirtualBox etc.)
- Install Linux/Unix or any related OS on the virtual machine.
- Have C compiler (ex: gcc).
- Keep using the environment while solving problems 1 to 5.

Or you can just start with repl.it, gcc in MacOS etc. without installing anything. But there will be no additional points.

Please remember that this is optional.

Problem 1

- Study the given code generating_n_threads.c that creates N threads. Each thread executes `thread_function()`.
- Create new function `thread_counting(void* argument)` that counts numbers (up to 30) and use it when creating threads in main().

```
thr_id = pthread_create(&threads[index], NULL,
thread_counting, &init_value[index]); // Use
this for Problem 1
```

- Set `NUM_THREADS` to 2
- Thus thread 1 counts odd numbers while thread 2 counts even numbers.

Output

```
main: creating thread 0
main: creating thread 1
pid: 48, tid: 4f084700, th_num: 0, number: 1
pid: 48, tid: 4e883700, th_num: 1, number: 2
pid: 48, tid: 4f084700, th_num: 0, number: 3
pid: 48, tid: 4e883700, th_num: 1, number: 4
pid: 48, tid: 4f084700, th_num: 0, number: 5
pid: 48, tid: 4e883700, th_num: 1, number: 6
pid: 48, tid: 4f084700, th_num: 0, number: 7
pid: 48, tid: 4e883700, th_num: 1, number: 8
pid: 48, tid: 4f084700, th_num: 0, number: 9
pid: 48, tid: 4e883700, th_num: 1, number: 10
pid: 48, tid: 4f084700, th_num: 0, number: 11
pid: 48, tid: 4e883700, th_num: 1, number: 12
pid: 48, tid: 4f084700, th_num: 0, number: 13
pid: 48, tid: 4e883700, th_num: 1, number: 14
pid: 48, tid: 4f084700, th_num: 0, number: 15
pid: 48, tid: 4e883700, th_num: 1, number: 16
pid: 48, tid: 4f084700, th_num: 0, number: 17
pid: 48, tid: 4e883700, th_num: 1, number: 18
pid: 48, tid: 4e883700, th_num: 1, number: 20
pid: 48, tid: 4f084700, th_num: 0, number: 19
pid: 48, tid: 4e883700, th_num: 1, number: 22
pid: 48, tid: 4f084700, th_num: 0, number: 21
pid: 48, tid: 4e883700, th_num: 1, number: 24
pid: 48, tid: 4f084700, th_num: 0, number: 23
pid: 48, tid: 4e883700, th_num: 1, number: 26
pid: 48, tid: 4f084700, th_num: 0, number: 25
pid: 48, tid: 4e883700, th_num: 1, number: 28
pid: 48, tid: 4f084700, th_num: 0, number: 27
pid: 48, tid: 4e883700, th_num: 1, number: 30
pid: 48, tid: 4f084700, th_num: 0, number: 29
```

Problem 2

- Set `NUM_THREADS` to `2`
  thus thread 0 still counts odd numbers while thread 1 counts even numbers.
- But use semaphore to print odd numbers first and even numbers later.

- Use binary semaphore
  (study the functions in semaphore.h.)

Output

```
main: creating thread 0
main: creating thread 1
pid: 1564, tid: 855d4700, th_num: 0, number: 1
pid: 1564, tid: 855d4700, th_num: 0, number: 3
pid: 1564, tid: 855d4700, th_num: 0, number: 5
pid: 1564, tid: 855d4700, th_num: 0, number: 7
pid: 1564, tid: 855d4700, th_num: 0, number: 9
pid: 1564, tid: 855d4700, th_num: 0, number: 11
pid: 1564, tid: 855d4700, th_num: 0, number: 13
pid: 1564, tid: 855d4700, th_num: 0, number: 15
pid: 1564, tid: 855d4700, th_num: 0, number: 17
pid: 1564, tid: 855d4700, th_num: 0, number: 19
pid: 1564, tid: 855d4700, th_num: 0, number: 21
pid: 1564, tid: 855d4700, th_num: 0, number: 23
pid: 1564, tid: 855d4700, th_num: 0, number: 25
pid: 1564, tid: 855d4700, th_num: 0, number: 27
pid: 1564, tid: 855d4700, th_num: 0, number: 29
pid: 1564, tid: 84dd3700, th_num: 1, number: 2
pid: 1564, tid: 84dd3700, th_num: 1, number: 4
pid: 1564, tid: 84dd3700, th_num: 1, number: 6
pid: 1564, tid: 84dd3700, th_num: 1, number: 8
pid: 1564, tid: 84dd3700, th_num: 1, number: 10
pid: 1564, tid: 84dd3700, th_num: 1, number: 12
pid: 1564, tid: 84dd3700, th_num: 1, number: 14
pid: 1564, tid: 84dd3700, th_num: 1, number: 16
pid: 1564, tid: 84dd3700, th_num: 1, number: 18
pid: 1564, tid: 84dd3700, th_num: 1, number: 20
pid: 1564, tid: 84dd3700, th_num: 1, number: 22
pid: 1564, tid: 84dd3700, th_num: 1, number: 24
pid: 1564, tid: 84dd3700, th_num: 1, number: 26
pid: 1564, tid: 84dd3700, th_num: 1, number: 28
pid: 1564, tid: 84dd3700, th_num: 1, number: 30
```

Problem 3

- Calculate the summation from 1 to N.
- The given code ‘calsum_code.c’ computes the summation from 1 to N with single thread.
- Please complete thread_function() to have calsum_MultiThread() work.
- After implementation of thread_function(), please uncomment the statements in main() to compare the result and speed between calsum() and calsum_MultiThread().
- Try with different `N` and `NUM_THREADS` and have discussion about the results.

Problem 4

- Implement bounded buffer problem using semaphore.
- Conditions
  - `int buffer[10];` // `buffersize` is 10 and buffer is global variable
  - There is only one producer and one consumer.
  - `Producer()`
    - At every 200msec,
    - generates random positive integer and add to the `buffer`.
    - Then print current state of `buffer` (numbers and in/out indices)
    - Also print current state of two semaphore `full` and `empty`.
    - After producing 20 numbers, `producer` terminates.
  - `Consumer()`
    - At every 170msec,
    - removes one integer from the `buffer`.
    - Then print current state of `buffer` (numbers and in/out indices)
    - Also print current state of two semaphore `full` and `empty`.
    - After reading 20 numbers, `consumer` terminates.
- Guidelines
  - Use three semaphore; `full`, `empty`, `mutex`
  - Use `usleep()` for delaying producer and consumer.

How to submit

- Download the template (.doc) for homework2 from class room in HisNet.
- Complete the report (problem 0 to 4), codes should be included in the report.
- Generate pdf file.
  Ex) HW02_20111245_HongGildong
  Ex) HW02_20111245_홍길동

- Submit the pdf file through HisNet.

- **Deadline: 6.1 (Saturday), 23:00**