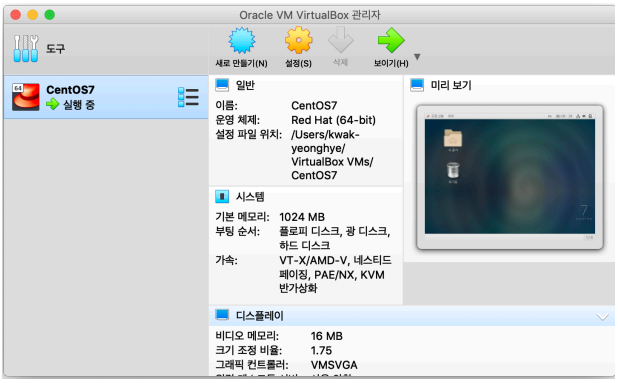
Operating systems	Student number	21700034
Homework 2	Name	곽영혜

☐ Summary of your achievement

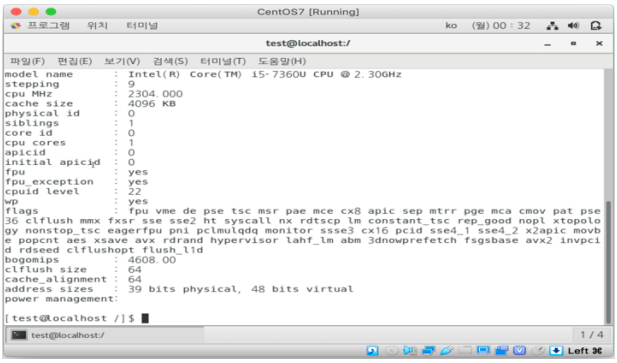
	Complete? (Yes / No)
Problem 0	yes
Problem 1	yes
Problem 2	yes
Problem 3	yes
Problem 4	no

☐ Problem 0 – if you say Yes

- Virtual Machine Name? VirtualBox(Oracle VM)
- OS Name and kernel number? Linux version 3.10.0-957
- Screenshot of OS running on the virtual machine



- Screenshot of checking Num. of CPUs / Cores

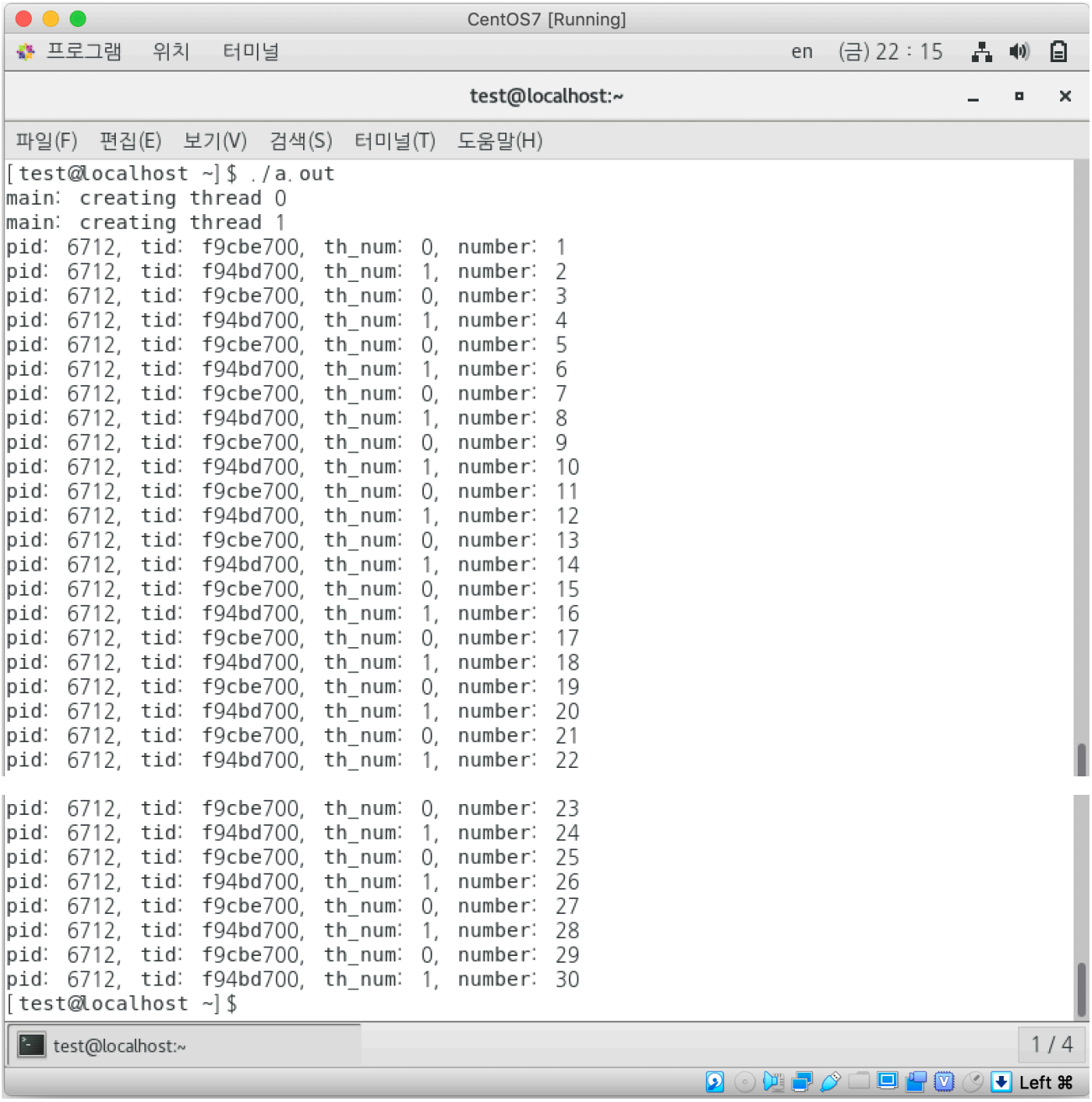


☐ Problem 0 – if you say No

- Please write the environment you used (OS and Compiler, or Repl.it):
- Screenshot of your environment.

☐ Problem 1

- Screenshot of the result



- Your code

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

#define NUM_THREADS 2 // number of threads

// ****** You implement this function ******
void* thread_counting(void* argument) {
  pid_t pid = getpid();          // process id
  pthread_t tid = pthread_self(); // thread id
  unsigned int ui_pid = (unsigned int)pid;
  unsigned int ui_tid = (unsigned int)tid;
  int i=1;

  int th_num = *((int*) argument); // passed value

  while(i<31) {
    if(i%2 == 0 && th_num%2==1) printf("pid: %u, tid: %x, th_num: %d,
number: %d\n", ui_pid, ui_tid, th_num, i);
    if(i%2 == 1 && th_num%2==0) printf("pid: %u, tid: %x, th_num: %d,
number: %d\n", ui_pid, ui_tid, th_num, i);
    i++;
    sleep(1);
  }
}

int main(int argc, char** argv) {
  pthread_t threads[NUM_THREADS];
  int init_value[NUM_THREADS];
  int thr_id;
  int result_code;
  unsigned index;

  // create all threads one by one
  for (index = 0; index < NUM_THREADS; index++) {
    init_value[index] = index;
    printf("main: creating thread %d\n", index);

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

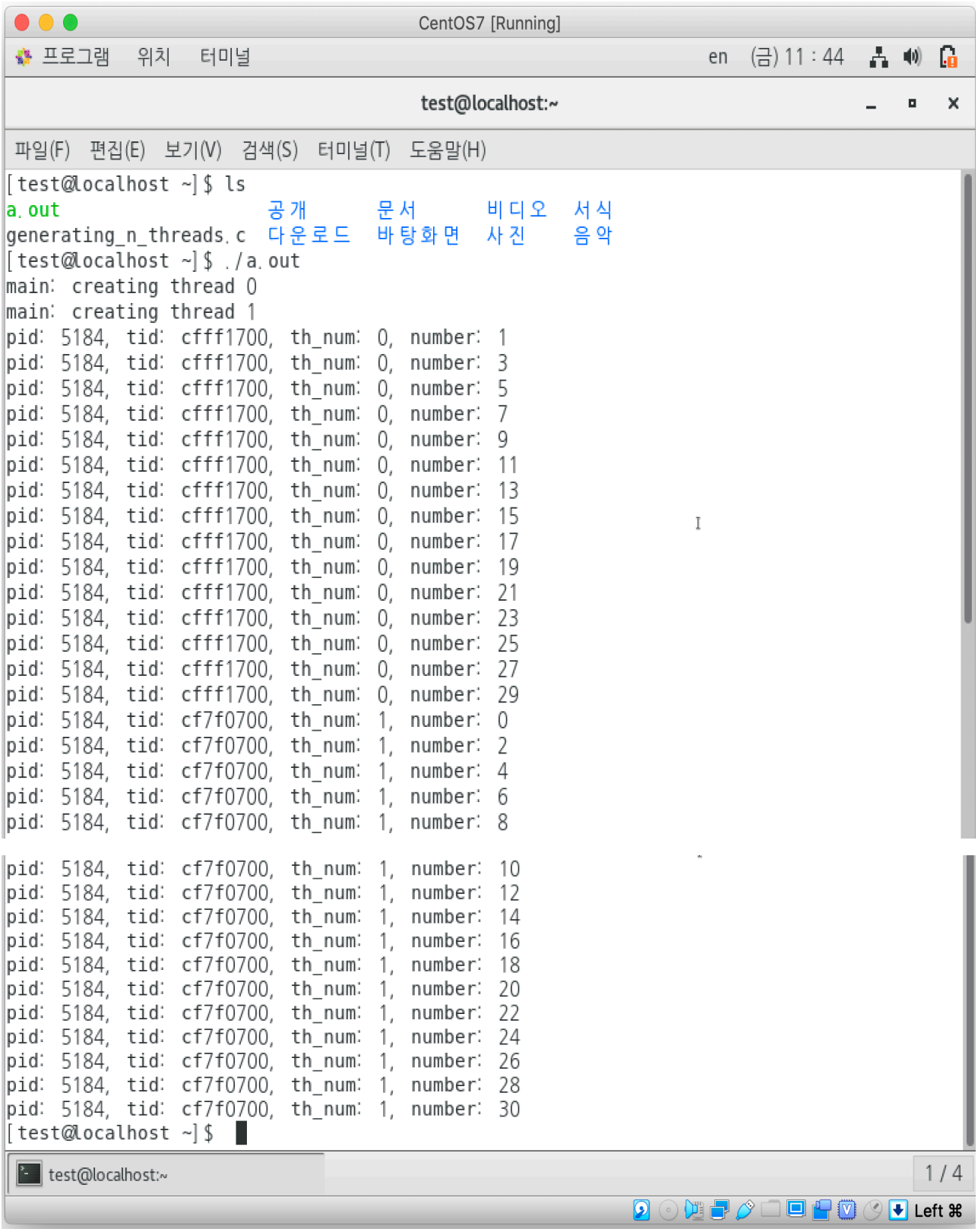
```
  }

  for (index = 0; index < NUM_THREADS; index++) {
    // block until ith thread completes
    result_code = pthread_join(threads[index], NULL);
    assert(!result_code); // if error_code (non-zero value) returned
than terminate.
  }

  exit(EXIT_SUCCESS);
}
```

- Discussion of what you learned and what was difficult.

첫번째 문제를 해결하기 위해 작성했던 코드를 돌려본 결과가 두번째 문제와 유사한 형태로 나오게 되어 왜 그런 것인지를 오류를 찾을려고 했으나 코드의 문제가 아닌, sleep으로 속도를 줄여주어 해결할 수 있음을 배우게 되었습니다.

☐ Problem 2

- Screenshot of the result



- Your code

```
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include <pthread.h>
#include <unistd.h>
#include <semaphore.h>

#define NUM_THREADS 2 // number of threads
sem_t semaphore;

// ****** You implement this function ******
void* thread_counting(void* argument) {
  //th_num = odd - odd / th_num = even - even
  pid_t pid = getpid();          // process id
  pthread_t tid = pthread_self(); // thread id
  unsigned int ui_pid = (unsigned int)pid;
  unsigned int ui_tid = (unsigned int)tid;
  int i=1;

  int th_num = *((int*) argument); // passed value

  while(i < 31) {
    if(th_num%2 == 1 && i%2 == 0) {
      sem_wait(&semaphore);
      printf("pid: %u, tid: %x, th_num: %d, number: %d\n", ui_pid,
ui_tid, th_num, i);
      sem_post(&semaphore);
    }
    else if(th_num%2 == 0 && i%2 == 1) {
      printf("pid: %u, tid: %x, th_num: %d, number: %d\n", ui_pid,
ui_tid, th_num, i);
      sem_post(&semaphore);
    }
    i++;
  }
}

int main(int argc, char** argv) {
  pthread_t threads[NUM_THREADS];
  int init_value[NUM_THREADS];
  int thr_id;
  int result_code;
  unsigned index;
```

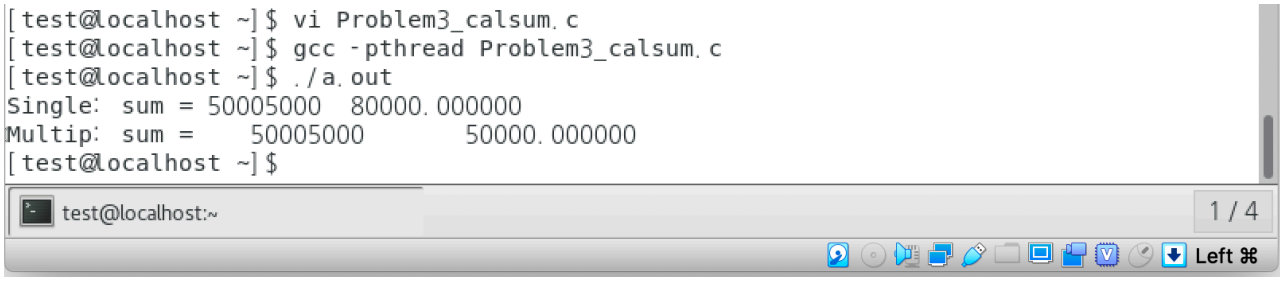
```
// create all threads one by one
  for (index = 0; index < NUM_THREADS; index++) {
    init_value[index] = index;
    printf("main: creating thread %d\n", index);

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

  for (index = 0; index < NUM_THREADS; index++) {
    // block until ith thread completes
    result_code = pthread_join(threads[index], NULL);
    assert(!result_code); // if error_code (non-zero value) returned
than terminate.
  }

  exit(EXIT_SUCCESS);
}
```

- Discussion of what you learned and what was difficult.

처음에는 repl.it에서 코드를 실행하였는데 thread가 생성되지 않는 경우가 너무 많아 코드에서 오류가 발생한 줄 알고 헤매어서 어려웠습니다. 그러나 환경 문제임을 알고 VM Linux 환경에서 동일한 프로그램을 작성 후, 실행하였더니 thread가 전부 생성되고 30까지 카운트가 됨을 확인할 수 있었습니다.

☐ Problem 3

- Screenshot of the result



- Your code

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

#define NUM_THREADS 2
#define N 1e4
long long f_sum=0, s_sum=0, sum = 0;

// matrix multiplication: c = a * b;
void calsum() {
  int i=1;
  for(i=1;i<=N;i++)
  {
    sum+=i;
    usleep(100);
  }
}

// Thread function
void* thread_function(void* argument) {
  int i;

  int th_num = *((int*) argument); // passed value

  if(th_num%2 == 0) {
    for(i=1;i<=N/2;i++) {
      f_sum+=i;
    }
  }
  else {
```

```
    for(i=N/2+1;i<=N;i++) {
      s_sum+=i;
      usleep(100);
    }
  }

  return NULL;
}

// Creates threads
void calsum_MultiThread() {
  pthread_t threads[NUM_THREADS];
  int init_value[NUM_THREADS];
  int thr_id;
  int result_code;
  unsigned index;

  // create all threads one by one
  for (index = 0; index < NUM_THREADS; index++) {
    init_value[index] = index;

    thr_id = pthread_create(&threads[index], NULL, thread_function,
&init_value[index]);
    // Use this for Problem 1
    assert(!thr_id);
  }

  for (index = 0; index < NUM_THREADS; index++) {
    // block until ith thread completes
    result_code = pthread_join(threads[index], NULL);
    assert(!result_code); // if error_code (non-zero value) returned
than terminate.
  }
  sum = f_sum + s_sum;
}

int main(int argc, char** argv) {
  float uSec_s, uSec_m;
  long long sum_s, sum_m;
  time_t start_t, end_t;

  // single thread
  start_t = clock();
  calsum();
  sum_s = sum;
  end_t = clock();
```

```
  uSec_s = 1000000*(float)(end_t - start_t)/CLOCKS_PER_SEC;

  printf("Single: sum = %lld\t%f\n", sum_s,uSec_s);

  sum = 0;
  // Multi thread
  start_t = clock();
  calsum_MultiThread();
  sum_m = sum;
  end_t = clock();
  uSec_m = 1000000*(float)(end_t - start_t)/CLOCKS_PER_SEC;

  printf("Multip: sum = %lld\t%f\n", sum_m,uSec_m);
}
```

- Discussion of what you learned and what was difficult.

Thread의 숫자에 따라서 어떤 부분의 합을 구해야 하는지 코드를 작성하는데 어려움을 겪었고 숫자들의 합을 다 구하기 위해 어떤 지점에서 thread의 속도를 늦춰줘야 하는지 판단하는데 어려움을 겪었습니다.

□ Problem 4 – bounded buffer problem

- Screenshot of the result

- Your code

- Discussion of what you learned and what was difficult.