```
/* CPSC 457 (Winter 2019)
 * Week 5 - 1
 * Tutorial 1 and 2
 * Sina Keshvadi
 *
 * Notes: No error handling!
 */
================================================
// Write and run this code and Check the result
// It's a simple code to increase a variable by one in every step

#include <stdio.h>

#define num_loops 20000000
long long sum = 0;

void counting(int offset)
{
        for (long long i=0; i<num_loops; i++)
                sum+=offset;
}

int main()
{
        counting(1);
        printf("Sum = %lld\n", sum);

        return(0);
}
================================================
use
time ./a.out
Remember the time of execution.
================================================
// Change  main function to this:

        int main()
        {
                counting(1);
                counting(-1);
                printf("Sum = %lld\n", sum);

                return(0);
        }

// then run and Check the result
================================================
use
time ./a.out
compare with the previous runtime
================================================
// use two threads to do the above example
#include <stdio.h>
#include <pthread.h>

#define num_loops 20000000
long long sum = 0;

void* counting(void *arg)
{
        int offset = *(int *) arg;
        for (long long i=0; i<num_loops; i++)
                sum+=offset;
        pthread_exit(NULL);
```

```
}

int main()
{
        pthread_t id1;
        int offset1 = 1;
        pthread_create(&id1, NULL, counting, &offset1);

        pthread_t id2;
        int offset2 = -1;
        pthread_create(&id2, NULL, counting, &offset2);

        //Wait for threads to finish
        pthread_join(id1, NULL);
        pthread_join(id2, NULL);

        printf("Sum = %lld\n", sum);

        return(0);
}

What is result? zero?!
run again and again?
Why you got different answers?
Change #define num_loops 100 and run again?
=================================================
A simple solution to critical section can be thought as shown below,

acquireLock();
Process Critical Section
releaseLock();
=================================================
// We have a critical section problem here
// we should find what the critical section is and then fix it
// with Lock Mechanism
// Here, a Critical section happens when you have shared data between multiple
// threads. rewrite your therad's function code according to below

pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;

void* counting(void *arg)
{
        int offset = *(int *) arg;
        for (long long i=0; i<num_loops; i++)
        {
                pthread_mutex_lock(&mutex);
                sum+=offset;
                pthread_mutex_unlock(&mutex);
        }
        pthread_exit(NULL);
}

=================================================
Run code again and again? Is it correct? Why?
use time ./a.out
Time is increased or decreased? WHY?
=================================================
// Run three times this code for create 10 random number in an array
#include <stdio.h>

int main()
{

    int myArray[10];
```

```
        for(int i=0; i<10; i++)
        {
                myArray[i] = rand() % 10;
        }

        for(int i=0; i<10; i++)
        {
                printf("%d \n", myArray[i]);
        }

    return 0;
}
================================================
// By using time as seed of the random generator,
// create real random numbers

#include <stdio.h>
#include <time.h>

int main()
{

    int myArray[10];
    srand ( time(NULL) );
        for(int i=0; i<10; i++)
        {
                myArray[i] = rand() % 10;
        }

        for(int i=0; i<10; i++)
        {
                printf("%d \n", myArray[i]);
        }

    return 0;
}
================================================
```