1A: Tiempo de procesamiento de SINHILOS.PY: 5.63 segundos, 5.62 segundos, 5.52 segundos

Tiempo de procesamiento de CONHILOS.PY: 4.43 segundos, 4.42 segundos, 4.53 segundos, 4.43 segundos

1B: Son iguales, pero con pequeñas diferencias entre sí en los números de los milisegundos

1C: SUMA\_RESTA.PY

1) 0.417

2) 0.465

3) 0.349

4) 0.269

5) 0.32

6) 0.284

7) 0.256

8) 0.327

9) 0.297

10) 0.28

SIN EL #

1) 0.385

2) 0.267

3) 0.273

4) 0.298

5) 0.281

6) 0.358

7) 0.275

8) 0.293

9) 0.278

10) 0.285

Lo que paso fue que, con los comentarios, al inicio el programa tardaba un poco más en correr hasta que después llego hasta el rango de 200 milisegundos. Pero después de descomentar el programa, el procesamiento del programa no subió a más de 385 milisegundos por lo que se puede decir que se aceleró un poco.

ACT 2

2A)

#include <pthread.h>

#include <stdio.h>

#include <stdlib.h>

#define NUMBER\_OF\_THREADS 2

#define CANTIDAD\_INICIAL\_HAMBURGUESAS 20

int cantidad\_restante\_hamburguesas = CANTIDAD\_INICIAL\_HAMBURGUESAS;

int turno = 0;

while(turno!=(int)tid);

void \*comer\_hamburguesa(void \*tid)

{

while (1 == 1)

{

// INICIO DE LA ZONA CRÍTICA

if (cantidad\_restante\_hamburguesas > 0)

{

printf("Hola! soy el hilo(comensal) %d , me voy a comer una hamburguesa ! ya que todavia queda/n %d \n", (int) tid, cantidad\_restante\_hamburguesas);

cantidad\_restante\_hamburguesas--; // me como una hamburguesa

turno = (turno + 1)% NUMBER\_OF\_THREADS;

}

else

{

printf("SE TERMINARON LAS HAMBURGUESAS :( \n");

pthread\_exit(NULL); // forzar terminacion del hilo

}

// SALIDA DE LA ZONA CRÍTICA

}

}

int main(int argc, char \*argv[])

{

pthread\_t threads[NUMBER\_OF\_THREADS];

int status, i, ret;

for (int i = 0; i < NUMBER\_OF\_THREADS; i++)

{

printf("Hola!, soy el hilo principal. Estoy creando el hilo %d \n", i);

status = pthread\_create(&threads[i], NULL, comer\_hamburguesa, (void \*)i);

if (status != 0)

{

printf("Algo salio mal, al crear el hilo recibi el codigo de error %d \n", status);

exit(-1);

}

}

for (i = 0; i < NUMBER\_OF\_THREADS; i++)

{

void \*retval;

ret = pthread\_join(threads[i], &retval); // espero por la terminacion de los hilos que cree

turno = (turno + 1)% NUMBER\_OF\_THREADS

}

pthread\_exit(NULL); // como los hilos que cree ya terminaron de ejecutarse, termino yo tambien.

}

2B)

