Práctica 1, Ejercicio 6 Introducción:

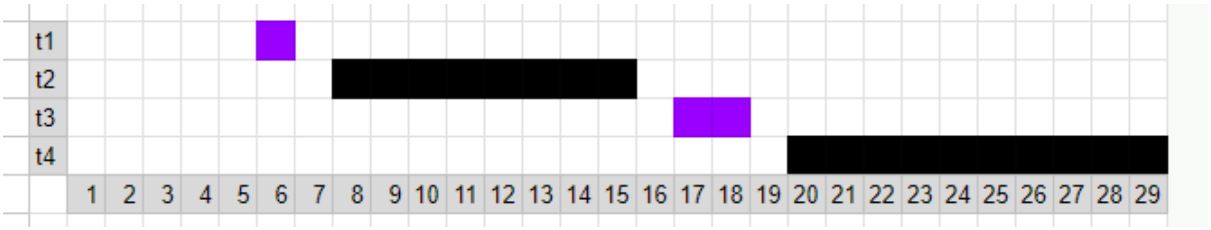
```
int main(){
    int vector[] = {1,10};
    float sum = 0;
    int cant;
    float prom;
    for (int i = 0; i < (sizeof(vector)/ sizeof(vector[0])); i++){
        sum += vector[i];
        cant = i + 1;
    }
    prom = sum/cant;
    cout << prom;
}
```

Práctica 1, Ejercicio 5 scheduling:

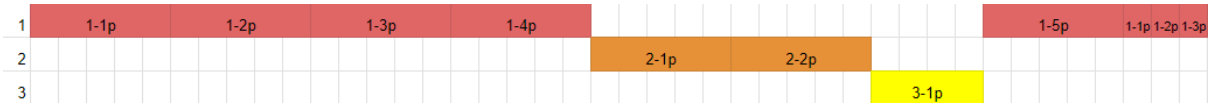
Q=10

Context Switch=1

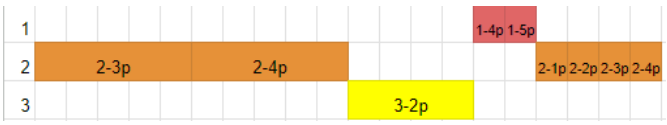
Tarea	Duración	Tiempo de llegada
T1	1	5
T2	8	6
T3	2	7
T4	10	9



Práctica 1, Ejercicio 17 scheduling:



(se corta porq no entra, pero sigue abajo)



- Cual es el primer proceso en terminar?
  Termina primero 3-1p.
- Cual es el tiempo promedio?
  Hay un total de 20 procesos y 124 unidade, promedio = 6,2 u.
- Cual es el tiempo total?
  124 unidades.
- Que habría pasado si todos los procesos de las queues 1 y 3, hubieran cambiado de lugar?
  La queue 1 terminaba antes.

Práctica 2, Ejercicio 8:

El código crea un total de 6 procesos. En las dos iteraciones del bucle for, se crean dos procesos por iteración, lo que da un total de 4 procesos creados. Luego, cada uno de estos procesos hijos vuelve a crear un proceso hijo adicional, lo que da un total de 2 procesos adicionales. Por lo tanto, en total se han creado 6 procesos.

El árbol de procesos creado puede verse como sigue:

```
  main process
       |
      / \
     /   \
child1  child2
  / \
 /   \
gc1  gc2
```

Una posible salida de consola al ejecutar el código podría ser:

```
Iteración número: 0
Iteración número: 0
Iteración número: 1
Iteración número: 1
Fin del proceso
Iteración número: 2
Iteración número: 2
```

Práctica 2, Ejercicio 17:

```
#include <unistd.h>
#include <iostream>
#include <cstdlib>

int main() {
    pid_t my_pid = getpid();
    pid_t parent_pid = getppid();

 if (my_pid > parent_pid) {
        exit(9);
    } else {
        std::string command = "./" + std::string(__FILE__);
        system(command.c_str());
 }
    return 0;
}
```

Práctica 3, Ejercicio 5:

El valor mínimo que puede obtener x es 3 y el valor máximo es 6.

Ambos bucles se ejecutan tres veces, lo que significa que el primer bucle aumenta "x" en un total de 6 unidades (3 iteraciones x 2 incrementos por iteración), y el segundo bucle lo disminuye en un total de 3 unidades (3 iteraciones x 1 decremento por iteración). Por ende ¿el incremento neto es de 6 unidades y el decremento neto es de 3 unidades, el valor final de "x" será 6 - 3 = 3. Por lo tanto, el valor mínimo que puede alcanzar "x" es 3.

No es posible que el valor final sea 0, debido a que el valor inicial de "x" es 0 y el incremento total es mayor que el decremento total, el valor final de "x" nunca puede ser 0.

Práctica 3, Ejercicio 7:

Una posible traza en la que el programa termina sería la siguiente:

- Inicialmente, la variable global "n" se establece en 0.
- El hilo "T1" comienza su ejecución y entra en el bucle "while". Dado que el valor de "n" es 0, el hilo "T1" ingresa al bucle.
- Dentro del bucle, "n" se incrementa en 1 y se convierte en 1.
- El hilo "T1" verifica nuevamente la condición del bucle "while". Como "n" es igual a 1, la condición no se cumple y el hilo "T1" sale del bucle.
- El programa ha alcanzado su estado final con "n" igual a 1.
- El hilo "T1" termina su ejecución.

Una posible traza en la que el programa no termina sería la siguiente:

- El hilo "T1" comienza su ejecución y entra en el bucle "while". Dado que el valor de "n" es 0, el hilo ingresa al bucle.
- Dentro del bucle, "n" se incrementa en 1 y se convierte en 1.
- El hilo "T1" verifica nuevamente la condición del bucle "while" y se mantiene dentro del bucle ya que "n" es igual a 1.
- El hilo "T2" comienza su ejecución y entra en su propio bucle "while". Dado que el valor de "n" es 1, el hilo "T2" ingresa al bucle.
- Dentro del bucle del hilo "T2", "n" se decrementa en 1 y se convierte en 0.
- El hilo "T2" vuelve a verificar la condición del bucle "while" y se mantiene dentro del bucle ya que "n" es igual a 0.
- Mientras tanto, el hilo "T1" sigue ejecutándose y repite los pasos 3-4, incrementando "n" en 1 y manteniéndose dentro del bucle.
- En este punto, los hilos "T1" y "T2" están atrapados en un ciclo infinito, alternando entre incrementar y decrementar "n" entre 1 y 0 respectivamente.
- El programa no termina ya que los hilos continúan ejecutando sus bucles en un ciclo infinito, sin alcanzar una condición que les permita salir de los bucles "while".

Práctica 3, Ejercicio 13:

```
semaphore_FA = threading.Semaphore(1)
semaphore_HE = threading.Semaphore(1)


count_A = 0
count_F = 0
count_E = 0
count_H = 0

def thread1_func():
    global count_A
    global count_F

    while True:
        semaphore_FA.acquire()
        print('A', end=' ')
        print('B', end=' ')
        print('C', end=' ')
        print('D')

        count_A += 1
        semaphore_FA.release()
        semaphore_HE.acquire()
        count_F += 1
        semaphore_HE.release()

def thread2_func():
    global count_F
    global count_E

    while True:
        semaphore_FA.acquire()

        if count_F <= count_A:
            print('E', end=' ')
            print('F', end=' ')
            print('G')

            count_F += 1

        semaphore_FA.release()

def thread3_func():
    global count_E
    global count_H
```

```
while True:
    semaphore_HE.acquire()

    if count_H <= count_E:
        print('H', end=' ')
        print('I')

        count_H += 1

    semaphore_HE.release()
```