Lesther Kevin Federico López Miculax

2021110897

Pr-2.1 Temporizadores

```
#include <iostream>
#include <queue>

template <class T, class K>
class PriorityQueue {
public:
   void add(T info, K key) {
      queue.emplace(key, info);
   }

   T get(K key) {

      return queue.top().second;
   }

   T head() {
      return queue.top().second;
   }

   T pop() {
      T info = queue.top().second;
      queue.pop();
      return info;
   }
```

```
private:
    std::priority_queue<std::pair<K, T>> queue;
};



int timer_stop() {

    return 0;
}

int timer_start(long time) {

    return 0;
}

long get_pid() {

    return 12345;
}

long send_signal(long pid, long signal) {

    return 0;
}

long get_time() {

    return 0;
}
```

```
PriorityQueue<TimerRequest, long> timer_queue;


void timer_interrupt_service() {
    if (!timer_queue.head()) {
        return;
    }

    TimerRequest current_request = timer_queue.pop();
    long current_time = get_time();

    if (current_time >= current_request.expiry_time) {
        send_signal(current_request.pid, current_request.signal);
    }

    if (timer_queue.head()) {
        TimerRequest next_request = timer_queue.head();
        long time_to_next_expiry = next_request.expiry_time - current_time;
        timer_start(time_to_next_expiry);
    }
}

void timer_create(long timems, long signal) {
    long pid = get_pid();
    long current_time = get_time();
    long expiry_time = current_time + timems;

    TimerRequest request = {pid, signal, expiry_time};
```

```
    timer_queue.add(request, expiry_time);
}

int main() {

    timer_create(1000, 10);
    timer_create(200, 40);



    timer_interrupt_service();

    return 0;
}
```