Homework 1:

You just landed a job as the chief software engineer at a high tech company. This company processes certain type of transactions. However, due to system delays the number of transactions that can be processed in a given time interval. Your job is to write a routine that will enforce this policy. Specifically, you have to implement a class "Dispatcher". The following is a skeleton for this class:

```
class Dispatcher
{
public:
  Dispatcher (unsigned int numberOfRequests, unsigned int time_interval);
  void dispatch_requests(unsigned long delay,  unsigned int request_count);
private:
  bool enable_request(); /* implement */
private:
  /* implement */
};

Dispatcher::Dispatcher(unsigned int numberOfRequests,unsigned int time_interval)
{
/*
numberOfRequests: the maximum number of requests allowed for a given time interval
time_interval: the value of the specific time interval in seconds.
For example:
Dispatcher(3,2) means that the Dispatcher can allow a maximum of 3 requests in any 2-second time interval
*/
}
```

The dispatch_request() function is provided for you:

```
void Dispatcher::dispatch_request(unsigned long delay, unsigned int request_count)

void
Dispatcher::dispatch_requests(unsigned long delay, unsigned int request_count)
{
/*
delay: number of seconds to wait before dispatching the tasks
request_count: number of task to dispatch
*/
  sleep(delay); // wait "delay" seconds before dispatching the tasks.
  for (unsigned int k=0; k<request_count; k++)
    {
    if (enable_request() == true)
      cout << "Request " << k<< " is being processed" << endl;
    else
      cout << "Request " << k<< " has been rejected" << endl;
    }
```

}

Your job is to implement the enable_request() function. This function returns *true* if the task can be dispatched, *false* otherwise.

The function get_current_time_in_secs() is provided to help you compute the elapsed time between two events. For example:

```
unsigned long begin = get_current_time_in_secs();
..... /* processing */
unsigned long end = get_current_time_in_secs();

elapsed_time = end-begin;
```