```
CC = gcc

CFLAGS += -Wall
CFLAGS += -g

SUBMIT_FILENAME = Apoio_T1.tar.gz

BINS = vector_stat_seq

all: $(BINS)

vector_stat_seq: vector_stat_seq.o
vector_stat_seq.o: vector_stat_seq.c


clean:
	$(RM) $(BINS) *.o $(SUBMIT_FILENAME)

submit: clean
	tar -czvf $(SUBMIT_FILENAME) *
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>


/**
 * Generates a random number between min and max.
 * The number sequence generated by the fuction rand is always the
 * same because it uses the defaulta seed 0.
 */
int get_random (int min, int max)
{
    return rand() % (max - min + 1) + min;
}

/**
 * Starts the vector with random values belonging to the interval [min..max]
 */
void vector_init_rand (int v[], long dim, int min, int max)
{
    for (long i = 0; i < dim; i++) {
        v[i] = get_random(min, max);
    }
}

/**
 * Gets the subvector of values in the range [min..max]
 *
 * returns the number of values store in subvector sv
 */
int vector_get_in_range (int v[], int v_sz, int sv[], int min, int max)
{
    long count = 0;

    for (long i = 0; i < v_sz; i++) {
        if (v[i] >= min && v[i] <= max) {
            sv[count++] = v[i];
        }
    }
    return count;
}


// Define o limite superior para a geração de valores aleatórios
#define LOWER_LIMIT       0
#define UPPER_LIMIT     100


int main (int argc, char *argv[])
{
    long values_sz = 256*1024*1024L;

    if (argc > 1) {
        values_sz = atol(argv[1]);
    }

    printf("Initializing a vector of %ld bytes\n", values_sz);

    // allocate a vector of initial values
    int *values = malloc(sizeof(int) * values_sz);
    if (values == NULL) {
        fprintf(stderr, "Erro malloc\n");
        return -1;
    }

    // allocate a subvector where will be store values in a interval of values
    int *subvalues = malloc(sizeof(int) * values_sz);
    if (subvalues == NULL) {
        fprintf(stderr, "Erro malloc\n");
        return -1;
    }

    // initiate initial array of values
    vector_init_rand(values, values_sz, LOWER_LIMIT, UPPER_LIMIT);

    struct timeval t1,t2;
    gettimeofday(&t1, NULL);

    // start of code to evaluate

    long count     = 0;
```

```
    int values_min = 50;
    int values_max = 100;

    count = vector_get_in_range(values, values_sz, subvalues, values_min, values_max);

    // end of code to evaluate

    gettimeofday(&t2, NULL);
    long elapsed = ((long)t2.tv_sec - t1.tv_sec) * 1000000L + (t2.tv_usec - t1.tv_usec);
    long sec = elapsed / (long)1e6;
    long aux = elapsed % (long)1e6;
    long mil = aux / (long)1e3;
    long mic = aux % (long)1e3;


    printf ("Elapsed time = %lds%ld,%ldms\n", sec, mil, mic);

    printf("Values between [%d..%d]: %ld\n", values_min, values_max, count);

    return 0;
}
```