EJERCICIO 1

Sistema operativo: Ubuntu 18.04.1 LTS
Compilador usado: ii g++ 4:7.3.0-3ubuntu2 amd64 GNU C++ compiler

a) Crear un fichero ordenacion.cpp con el programa completo para realizar una ejecución del algoritmo.

```
#include <iostream>
#include <ctime>    // Recursos para medir tiempos
#include <cstdlib>  // Para generación de números pseudoaleatorios

using namespace std;

void ordenar (int *v, int n){
  for (int i=0; i<n-1; i++)
    for (int j=0; j<n-i-1; j++){
      if (v[j] > v[j+1]){
        int aux = v[j];
        v[j] = v[j+1];
        v[j+1] = aux;
      }
    }
}

void sintaxis()
{
  cerr << "Sintaxis:" << endl;
  cerr << "  TAM: Tamaño del vector (>0)" << endl;
  cerr << "  VMAX: Valor máximo (>0)" << endl;
  cerr << "Se genera un vector de tamaño TAM con elementos aleatorios en [0,VMAX[" << endl;
  exit(EXIT_FAILURE);
}

int main(int argc, char * argv[])
{
  // Lectura de parámetros
  if (argc!=3)
    sintaxis();
  int tam=atoi(argv[1]);     // Tamaño del vector
  int vmax=atoi(argv[2]);    // Valor máximo
  if (tam<=0 || vmax<=0)
    sintaxis();

  // Generación del vector aleatorio
  int *v=new int[tam];       // Reserva de memoria
  srand(time(0));            // Inicialización del generador de números pseudoaleatorios
  for (int i=0; i<tam; i++)  // Recorrer vector
    v[i] = rand() % vmax;    // Generar aleatorio [0,vmax[

  clock_t tini;    // Anotamos el tiempo de inicio
  tini=clock();
```

```
  ordenar(v,tam); // de esta forma forzamos el peor caso

  clock_t tfin;    // Anotamos el tiempo de finalización
  tfin=clock();

  // Mostramos resultados
  cout << tam << "\t" << (tfin-tini)/(double)CLOCKS_PER_SEC << endl;

  delete [] v;     // Liberamos memoria dinámica
}
```

b) Crear un script ejecuciones_ordenacion.csh en C-Shell que permite ejecutar varias veces el programa anterior y generar un fichero con los datos obtenidos.

```
#!/bin/csh
@ inicio = 100
@ fin = 30000
@ incremento = 500
set ejecutable = ordenacion
set salida = tiempos_ordenacion.dat

@ i = $inicio
echo > $salida
while ( $i <= $fin )
  echo Ejecución tam = $i
  echo `./{$ejecutable} $i 10000` >> $salida
  @ i += $incremento
end
```

Luego lo he ejecutado y he obtenido en el fichero “tiempos_ordenacion.dat” lo siguiente:

```
100 4.1e-05
600 0.000936
1100 0.002628
1600 0.007172
2100 0.010154
2600 0.013621
3100 0.019354
3600 0.027235
4100 0.041155
4600 0.047436
5100 0.062821
5600 0.071603
6100 0.094536
6600 0.108671
7100 0.124035
7600 0.14224
8100 0.163256
8600 0.188868
9100 0.209572
9600 0.235787
```

10100 0.26871
10600 0.312005
11100 0.332003
11600 0.351224
12100 0.390672
12600 0.441777
13100 0.466953
13600 0.517464
14100 0.542354
14600 0.592751
15100 0.656582
15600 0.699226
16100 0.710984
16600 0.797106
17100 0.830137
17600 0.881512
18100 0.929104
18600 1.02322
19100 1.06077
19600 1.10697
20100 1.18979
20600 1.21649
21100 1.29898
21600 1.36078
22100 1.41452
22600 1.48412
23100 1.52479
23600 1.58051
24100 1.65128
24600 1.73633
25100 1.79327
25600 1.8498
26100 1.93544
26600 2.00404
27100 2.10235
27600 2.16979
28100 2.26685
28600 2.37293
29100 2.45454
29600 2.53382

c) Usar gnuplot para dibujar los datos obtenidos en el apartado previo.