IN3200/IN4200 Exercise Set 1

Exercise 1

- Write a C program to verify that the limit of $1 - \frac{1}{2^2} + \frac{1}{2^4} - \frac{1}{2^6} + \ldots$ is $\frac{4}{5}$.
- Write a C program that allocates a 1D array of length n which is prescribed at runtime. You are supposed to first assign the values of the array with random numbers, and then find the maximum and minimum values. (You can use e.g. the `rand` function from `stdlib.h`.)
- When assigning values to the entries of a $m \times n$ matrix, it is common to use a nested `for`-loop with the outer index looping over the rows and the inner index looping over the columns. Does it matter if the sequence of these two loops is swapped? Write a C program to test your hypothesis.
- Write a C program that allocates a 3D array of dimension (n_x, n_y, n_z). A 1D underlying contiguous storage should be used. Assign the values of the 3D array, such as `u[i][j][k]=i*n_y*n_z+j*n_z+k`. Deallocate the 3D array at the end of the program.

Exercise 2

- Write a C program that reads from a data file containing one day's temperature measurements of the following format:

```
00:05  -0.1
00:21   0.1
00:29  -0.2
...
```

Find out the highest and lowest temperatures and when they occurred. Compute also the average temperature and the associated standard deviation.

- Extend the `smooth` function (shown in the slides of the first lecture) to be applicable to a 2D array, for which the numerical formula is

$$v_{i,j}^{\text{new}} = v_{i,j} + c\,(v_{i-1,j} + v_{i,j-1} - 4v_{i,j} + v_{i,j+1} + v_{i+1,j})$$

Exercise 3

- The following two functions implement the famous quicksort: (see `http://alienryderflex.com/quicksort/`)

```
void swap(int *a, int *b)
{
  int t=*a; *a=*b; *b=t;
}
void sort(int arr[], int beg, int end)
{
  if (end > beg + 1) {
    int piv = arr[beg], l = beg + 1, r = end;
    while (l < r) {
      if (arr[l] <= piv)
        l++;
      else
        swap(&arr[l], &arr[--r]);
    }
    swap(&arr[--l], &arr[beg]);
    sort(arr, beg, l);
    sort(arr, r, end);
  }
}
```

Modify the `sort` function such that instead of directly sorting the array `arr`, we keep it as is but produce a so-called permutation vector `perm`. The purpose is that `arr[perm[0]]`, `arr[perm[1]]`, ..., `arr[perm[n-1]]` is an ordered series.