HBM516E – Scientific Visualization

HOMEWORK 1

Remarks:
Write the code yourself. ***Cheating is strictly forbidden.***
For each problem write your code in the function format and give the names of the functions as problem numbers, for example for the solution of problem1:

```
def problem1(input):
  return something
```

Put the codes for all problems into one file (jupter notebook file) and name that file using your student username in the following format: badays_hmb516e_homework1.ipynb. The notebook file should definitely contain the outputs of the functions, if applicable. Sample solution file (sample_solution.ipynb) is given to you to show how to organize your solutions.

Questions

1. **(50pt)** Write a function which compresses a NetCDF file. Your function should work on any NetCDF file (independent from the structure of NetCDF file). Each variable in input file must be converted to “short” (NC_SHORT) data type before writing to output NetCDF file. You can work on “input.nc” file given to you.

   Your function should be called like below. The function has two inputs: the name of the input file to be compressed and the name of compressed file.

```
problem1("input.nc","input_compressed.nc")
```

   Hint: The file can be compressed (“lossy”, original input values cannot be reproduced) using **add_offset** and **scale_factor** variable attributes. The attributes **scale_factor** and **add_offset** can be used together to provide simple data compression to store low-resolution floating-point data as small integers in a netCDF dataset. When scaled data are written, the application should first subtract the offset and then divide by the scale factor, rounding the result to the nearest integer to avoid a bias caused by truncation towards zero (see unpacking algorithm). In general, visualization software handles unpacking automatically.

The packing algorithm is:

$$\begin{aligned}
\text{scale\_factor} &= (\text{max} - \text{min})/\text{ndrv} \\
\text{add\_offset} &= (\text{min} + \text{max})/2 \\
\text{pck} &= (\text{upk} - \text{add\_offset})/\text{scale\_factor} \\
&= \frac{\text{ndrv} \times [\text{upk} - (\text{min} + \text{max})/2]}{\text{max} - \text{min}}
\end{aligned}$$

The unpacking algorithm is:

$$\begin{aligned}\text{upk} &= \text{scale\_factor} \times \text{pck} + \text{add\_offset} \\ &= \frac{\text{pck} \times (\max - \min)}{\text{ndrv}} + \frac{\min + \max}{2}\end{aligned}$$

For NC_SHORT ndrv is **2^{15}-1**. For more information read following documents:

http://nco.sourceforge.net/nco.pdf

http://www.unidata.ucar.edu/software/netcdf/docs/netcdf/Attribute-Conventions.html

https://www.ncl.ucar.edu/Document/Functions/Contributed/pack_values.shtml

2. **(50pt)** Use the same file given in previous question and write another Python script to create new netCDF file with reduced data. In this case, the file with reduced data will contain the original data but every Nth grid points will be skipped in each dimension.

   For example:

   The original file dimensions are 52, 52, 200 in x, y and time respectively. If N (must be a parameter in the script) is given as 2, then new dimension size in each direction will be 26x26x100 and the data in the reduced netCDF file will include the data of 1, 3, 5, 7, …, N-1 indexes in the original netCDF data

   Your function should be called like below:

```
problem2("input.nc","reduced_input.nc",2)
```