L3. Concurrent programming for graphics processors

Data files are reused from L1 and L2, however, minimum amount of elements is 250 (elements can be duplicated). An example of data file:


```
{
  "students": [
    {"name": "Antanas", "year": 1, "grade": 6.95},
    {"name": "Kazys", "year": 2, "grade": 8.65},
    {"name": "Petras", "year": 2, "grade": 7.01},
    {"name": "Sonata", "year": 3, "grade": 9.13},
    {"name": "Jonas", "year": 1, "grade": 6.95},
    {"name": "Martynas", "year": 3, "grade": 9.13},
    {"name": "Artūras", "year": 2, "grade": 7.01},
    {"name": "Vacys", "year": 2, "grade": 8.65},
    {"name": "Robertas", "year": 3, "grade": 6.43},
    {"name": "Mykolas", "year": 1, "grade": 6.95},
    {"name": "Aldona", "year": 3, "grade": 9.13},
    {"name": "Asta", "year": 2, "grade": 7.01},
    {"name": "Viktoras", "year": 2, "grade": 8.65},
    {"name": "Artūras", "year": 5, "grade": 8.32},
    {"name": "Vytas", "year": 3, "grade": 7.85},
    {"name": "Jonas", "year": 1, "grade": 6.95},
    {"name": "Zigmas", "year": 3, "grade": 9.13},
    {"name": "Artūras", "year": 2, "grade": 7.01},
    {"name": "Simas", "year": 3, "grade": 6.43}
  ]
}
```


Data is read from the file, distributed between a selected amount of GPU threads (amount of data must **not divide equally** by the total amount of threads; at least two blocks must be used, block size must divide by 32). A thread computes a selected result of **text type** for each element of its data set and, if it matches the selected condition, writes it to an array that is shared between all threads (it is allowed to set a fixed amount of symbols for each item and fill missing characters with spaces). If some of the elements do not match the criterion, there must not be any empty words in the result array — all results have to be written in the first available position of the array at the time of writing.

Program must be implemented using CUDA threads and CUDA memory management functions.

Example of a result file: (name is changed to capital letters, year is concatenated and a grade between A and F is determined from the grade in the file; only elements with first letter that is alphabetically further than P are selected):

```
SONATA-3A
VACYS-2B
ROBERTAS-3D
VIKTORAS-2B
VYTAS-3C
ZIGMAS-3A
SIMAS-3D
```

Test

Test is done on Moodle during the lab lecture. Test questions are from theory lectures "Memory management in C and C++", "Concurrent programming in CUDA" and "Thrust". Test duration — 20 minutes.

L3 grading

- L3a — 6 points (lecture 15, week 16 by AIS)
- Test — 4 points (lecture 16, week 17 by AIS)

Programs have to be presented during lab lectures not later than the deadline, program (.cu), data and result files (.txt) have to be uploaded to Moodle before presentation.

In order to complete the lab, NVidia video card is required.

Installing CUDA on your PC with an NVidia card

CUDA can be downloaded from NVidia page. It is important to check which version of CUDA is supported by your GPU.

Working with class computers

Class computers have CUDA installed, but the computers have old video cards installed (6 series) and do not support the latest version of CUDA. For this reason CUDA projects have to be created using **Visual Studio 2012**.

CUDA server

CUDA server can be accessed via SSH at address `cudalab.int.ktu.lt`, login data is the same as MPI server's. In order to get your login details contact your labs lecturer. Server has CUDA 9.1 installed, for program compiling instructions see "Compiling CUDA on Linux".

Creating CUDA projects using Visual Studio

`File` -> `New Project` -> `NVIDIA` -> `CUDA 9.x`.

Program file extension - `.cu`.

In CUDA 9 in order to be able to use `printf` in your CUDA functions, set:

`Project` -> `Properties` -> `Configuration properties` -> `CUDA C / C++` -> `Device` -> `Code Generation: compute_20,sm_20`.

Compiling CUDA on Linux

NVCC compiler is used. Used is similar to gcc, e. g.:

```
nvcc program.cu -o program
```