CSCI 325 Spring 2022
Mini Project 2: Convolutional Layer

Objectives:
1. Filtering operation.
2. Bias operation.
3. Activation (Sigmoid).

Notes:

Rules:
This programming assignment will be graded on style and structure as well as correctness. Students are ALLOWED to use **standard CUDA C/C++ libraries**. There should be comments and indentation, variables and functions should have meaningful names, and the program should have a clear, simple structure.

The programming assignment must be individual work. Students MAY discuss it with your classmates for conceptual and logical understanding. But, **MUST NOT** share your solution (or source codes) with anyone, in person or electronically, until the end of the semester. Do not discuss the assignment online, or copy online solutions. If you find some codes online, **you MUST cite both the website domain and the source codes**; in other words it is strictly prohibited to directly copy some or most of the source codes online. (How to cite will be shown at the end of this file.) Your program will be run through an automated plagiarism checker. If you are found to have cheated on any single programming assignment, **you will receive a 0 on ALL the programming assignments, past and future**, so consider this carefully. **Late work** on this assignment will be accepted, with **a penalty of 20% per day**.

Write a program that:
```
// 1. forward pass for each image.
    // 2-1. call function kernel_conv_filter()
    // 2-2. call function kernel_conv_bias()
    // 2-3. call function kernel_conv_sigmoid()
```

Input Files:
Basically, not needed.

Programming requirements:
Verification part MUST be included.

Please submit only your **Conv1_FirstName_LastName.c file** on the moodle (For two students, **Conv1_FirstName1_LastName1_FirstName2_LastName2.c**)

Programming details

It is allowed to modify attributes or/and methods in the given class or/and in the given CUDA kernel functions (e.g., parameter numbers, types or/and names), if it is needed. However, all the descriptions in the following PAs will be based on the class. (**NOTE**: for more straightforward readability in computation/operations, some hard-coded array index numbers are intentionally used; it can be more generalized/refactorized after the correctness tests or using pointers).

- The class **Layer**:

```
class Layer{
    public:
    int M, N, O; // O: output, N: #feature, M: #params_per_feature
    float *pre_output, *output;
    float *weight, *bias;

    Layer(int M, int N, int O);
```

```
        ~Layer();

    };
```

- Write a function: main()
    - load data (in PA2)
    - forward pass

- Write a constructor: **Layer::Layer(int M, int N, int O);**
    - Initialization for attributes.
    - Initialization for CUDA memory management

- Write a destructor: **Layer::~Layer();**
    - Free CUDA memory

- Write a function: ***void forward_pass(double data[28][28]);***
    - call function ***kernel_conv_filter*** (float input[28][28], float pre_output[6][24][24],
      float weight[6][5][5]);
        - **The total task T = (6*5*5)*(24*24**)
        - ***kernel_conv_filter<<< X1, Y1 >>> (a, b, c);***

    - call function ***kernel_conv_bias***(float pre_output[6][24][24], float bias[6])
        - **The total task T =6*24*24**
        - ***kernel_conv_bias<<< X2, Y2 >>> (a, b);***

    - call function ***kernel_conv_sigmoid***(float pre_output[6][24][24], float output[6][24][24])
        - **The total task T =6*24*24**
        - ***kernel_conv_sigmoid<<< X3, Y3 >>> (a, b);***

Tips for Design and Implementation:
Test-driven design and implementation step by step.

Sample Output file:

At least SHOULD have the results, optional for more vigorous verification.
Constructor: weight and bias values are initialized to **-1.0f** for testing
input value has changed to floating type **-1.0f** for testing

```
After Filtering:
preact maxError = 0.000000 (asserted with 25.0)
After Bias:
preact maxError = 0.000000 (asserted with 24.0)
After Sigmoid:
output maxError = 0.000000 (asserted with 1.0)
```

How to cite both the website domain and the source codes:

Below the main function, please add the comments and the full domain address with the source codes like the example (without any white spaces or indentation for http(s)). If you refer to many sites, please add consecutively. (You don't have to add the sites for checking the signatures of C standard library functions).

```
int main(){
        // your codes
        return 0;
}
```

/*START_CITE

http://www.domain1.com (or https://xxx)

Referred code 1 (just example)

```
void myMemCpy(void *dest, void *src, size_t n)
{
   // Typecast src and dest addresses to (char *)
   char *csrc = (char *)src;
   char *cdest = (char *)dest;

   // Copy contents of src[] to dest[]
   for (int i=0; i<n; i++)
       cdest[i] = csrc[i];
}
```

https://www.domain2.com

Referred code 2

END_CITE*/