Title of the Assignment:

Write a CUDA Program for :

1. **Addition of two large vectors**
2. **Matrix Multiplication using CUDA C**

Objective of the Assignment:

Students should be able to perform Program for Addition of two large vectors & Matrix Multiplication using CUDA C

Prerequisite:

1. CUDA Concept
2. Matrix Multiplication
3. Vector Addition
4. How to execute Program in CUDA Environment

Theory:

What is CUDA?

CUDA (Compute Unified Device Architecture) is a parallel computing platform and programming model developed by NVIDIA. It allows developers to use the power of NVIDIA graphics processing units (GPUs) to accelerate computation tasks in various applications, including scientific computing, machine learning, and computer vision.CUDA provides a set of programming APIs, libraries, and tools that enable developers to write and execute parallel code on NVIDIA GPUs. It supports popular programming languages like C, C++, and Python, and provides a simple programming model that abstracts away much of the low-level details of GPU architecture. Using CUDA, developers can exploit the massive parallelism and high computational power of GPUs to accelerate computationally intensive tasks, such as matrix operations, image processing, and deep learning. CUDA has become an important tool for scientific research and is widely used in fields like physics, chemistry, biology, and engineering.

A. Steps for Addition of two large vectors using CUDA

1. Define the size of the vectors: In this step, you need to define the size of the vectors that you want to add. This will determine the number of threads and blocks you will need to use to parallelize the addition operation.

2. Allocate memory on the host: In this step, you need to allocate memory on the host for the two vectors that you want to add and for the result vector. You can use the C malloc function to allocate memory.

3. Initialize the vectors: In this step, you need to initialize the two vectors that you want to add on the host. You can use a loop to fill the vectors with data.

4. Allocate memory on the device: In this step, you need to allocate memory on the device for the two vectors that you want to add and for the result vector. You can use the CUDA function cudaMalloc to allocate memory.

5. Copy the input vectors from host to device: In this step, you need to copy the two input vectors from the host to the device memory. You can use the CUDA function cudaMemcpy to

copy the vectors.

6. Launch the kernel: In this step, you need to launch the CUDA kernel that will perform the addition operation. The kernel will be executed by multiple threads in parallel. You can use the <<>> syntax to specify the number of blocks and threads to use.

7. Copy the result vector from device to host: In this step, you need to copy the result vector from the device memory to the host memory. You can use the CUDA function cudaMemcpy to copy the result vector.

B. Steps for Matrix Multiplication using
CUDA Here are the steps for implementing matrix multiplication using CUDA C:

1. Matrix Initialization: The first step is to initialize the matrices that you want to multiply. You can use standard C or CUDA functions to allocate memory for the matrices and initialize their values. The matrices are usually represented as 2D arrays.

2. Memory Allocation: The next step is to allocate memory on the host and the device for the matrices. You can use the standard C malloc function to allocate memory on the host and the CUDA function cudaMalloc() to allocate memory on the device.

3. Data Transfer: The third step is to transfer data between the host and the device. You can use the CUDA function cudaMemcpy() to transfer data from the host to the device or vice versa.

4. Kernel Launch: The fourth step is to launch the CUDA kernel that will perform the matrix multiplication on the device. You can use the <<>> syntax to specify the number of blocks and threads to use. Each thread in the kernel will compute one element of the output matrix.

5. Device Synchronization: The fifth step is to synchronize the device to ensure that all kernel executions have completed before proceeding. You can use the CUDA function cudaDeviceSynchronize() to synchronize the device.

6. Data Retrieval: The sixth step is to retrieve the result of the computation from the device to the host. You can use the CUDA function cudaMemcpy() to transfer data from the device to the host.

7. Memory Deallocation: The final step is to deallocate the memory that was allocated on the host and the device. You can use the C free function to deallocate memory on the host and the CUDA function cudaFree() to deallocate memory on the device.

Execution of Program over CUDA Environment

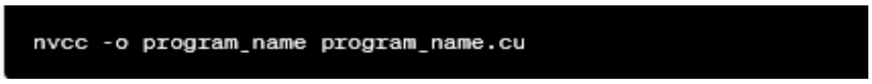
Here are the steps to run a CUDA program for adding two large vectors:

1. Install CUDA Toolkit: First, you need to install the CUDA Toolkit on your system. You can download the CUDA Toolkit from the NVIDIA website and follow the installation instructions provided.

2. Set up CUDA environment: Once the CUDA Toolkit is installed, you need to set up the CUDA environment on your system. This involves setting the PATH and LD_LIBRARY_PATH environment variables to the appropriate directories.

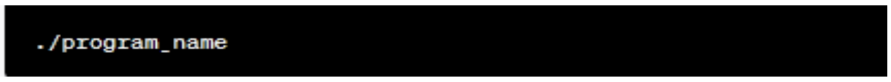
3. Write the CUDA program: You need to write a CUDA program that performs the addition of two large vectors. You can use a text editor to write the program and save it with a .cu extension.

4. Compile the CUDA program: You need to compile the CUDA program using the nvcc compiler that comes with the CUDA Toolkit. The command to compile the program is:



```
nvcc -o program_name program_name.cu
```

5. This will generate an executable program named program_name. Run the CUDA program: Finally, you can run the CUDA program by executing the executable file generated in the previous step. The command to run the program is: This will execute the program and perform the addition of two large vector



```
./program_name
```

Conclusion-

In this way we can implement Program for Addition of two large vectors & Matrix Multiplication using CUDA C