Question 2

The platform used to run experiments for this question is the **Discovery Cluster** using sbatch. More information on the code and how to run is provided in the Readme and in the log file.

Part A:

The table below shows the time elapsed in doing stencil computation using tiled and non-tiled implementation.

Total threads = grid_size * block_size	Tiled Implementation (microseconds)	Non - Tiled Implementation (microseconds)
(32, 32, 32) * (2, 2, 2)	383	9
(16, 16, 16) * (4, 4, 4)	406	9
(8, 8, 8) * (8, 8, 8)	406	9
(4, 4, 4) * (16, 16, 16)	417	1

- From the above table, the time taken by the tiled implementation is higher compared to the non-tile implementation.
- This is due to the additional time spent in creating shared memory and assigning values to the shared memory from the global memory.
- There is also additional time spent on assigning values in shared memory which have overlap with other threads in the block.
- The above mentioned additional time overshadows the benefits from exploiting spatial locality in the shared memory.

Part B:

Following factors can accelerate the performance on the GPU,

- The type of memory in which data is kept, how the data is set out, and the order in which it is retrieved, among other considerations, can all have a significant impact on bandwidth.
- The use of data type plays a considerable role for this implementation. Lesser the size of the data type used, less the time taken to send the data from host memory to device and less the time taken to create and fill shared memory.
- Minimizing data transmission between the host and the device is critical for overall program performance, even if this means running kernels on the GPU that do not show any speedup over running them on the host CPU. In short we could also avoid using unwanted variables and group the arrays.