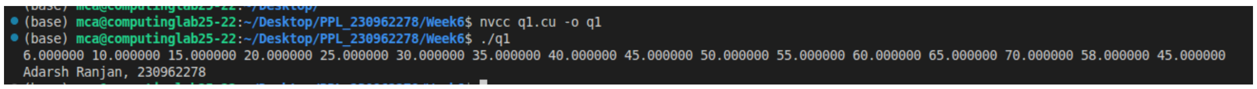
Week - 6

Q1:

```
#include <stdio.h>
#include <cuda_runtime.h>

__global__ void convolution1D(float *N, float *M, float *P, int width, int
mask_width)
{
   int i = blockIdx.x * blockDim.x + threadIdx.x;
   int half = mask_width / 2;
   if (i < width)
   {
       float sum = 0.0f;
       for (int j = 0; j < mask_width; j++)
       {
           int idx = i - half + j;
           if (idx >= 0 && idx < width)
               sum += N[idx] * M[j];
       }
       P[i] = sum;
   }
}

int main()
{
   int width = 16;
   int mask_width = 5;

   float h_N[16], h_M[5], h_P[16];

   for (int i = 0; i < width; i++)
       h_N[i] = i + 1;

   for (int i = 0; i < mask_width; i++)
       h_M[i] = 1.0f;

   float *d_N, *d_M, *d_P;
```

```
    cudaMalloc((void**)&d_N, width * sizeof(float));
    cudaMalloc((void**)&d_M, mask_width * sizeof(float));
    cudaMalloc((void**)&d_P, width * sizeof(float));

    cudaMemcpy(d_N, h_N, width * sizeof(float), cudaMemcpyHostToDevice);
    cudaMemcpy(d_M, h_M, mask_width * sizeof(float),
cudaMemcpyHostToDevice);

    int blockSize = 256;
    int gridSize = (width + blockSize - 1) / blockSize;

    convolution1D<<<gridSize, blockSize>>>(d_N, d_M, d_P, width,
mask_width);

    cudaMemcpy(h_P, d_P, width * sizeof(float), cudaMemcpyDeviceToHost);

    for (int i = 0; i < width; i++)
        printf("%f ", h_P[i]);
    printf("\nAdarsh Ranjan, 230962278");

    printf("\n");

    cudaFree(d_N);
    cudaFree(d_M);
    cudaFree(d_P);

    return 0;
}
```

Output:



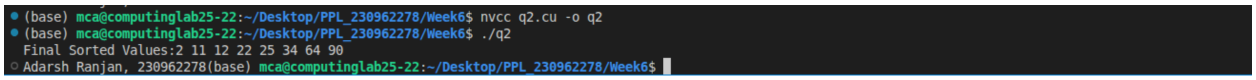
```
• (base) mca@computinglab25-22:~/Desktop/PPL_230962278/Week6$ nvcc q1.cu -o q1
• (base) mca@computinglab25-22:~/Desktop/PPL_230962278/Week6$ ./q1
  6.000000 10.000000 15.000000 20.000000 25.000000 30.000000 35.000000 40.000000 45.000000 50.000000 55.000000 60.000000 65.000000 70.000000 58.000000 45.000000
  Adarsh Ranjan, 230962278
```

Q2:

```
#include <stdio.h>
#include <cuda.h>

__global__ void findMinIndex(int *arr, int *minIdx, int n, int i)
{
   int tid = threadIdx.x;
   if (tid == 0)
   {
       int min_index = i;
       int j;
       for (j = i + 1; j < n; j++)
       {
           if (arr[j] < arr[min_index])
               min_index = j;
       }
       *minIdx = min_index;
   }
}

int main()
{
   int n = 8;
   int h_arr[8] = {64, 25, 12, 22, 11, 90, 34, 2};
   int *d_arr, *d_minIdx;
   int i, minIdx, temp;

   cudaMalloc((void**)&d_arr, n * sizeof(int));
   cudaMalloc((void**)&d_minIdx, sizeof(int));

   cudaMemcpy(d_arr, h_arr, n * sizeof(int), cudaMemcpyHostToDevice);

   for (i = 0; i < n - 1; i++)
   {
       findMinIndex<<<1, 1>>>(d_arr, d_minIdx, n, i);
       cudaMemcpy(&minIdx, d_minIdx, sizeof(int), cudaMemcpyDeviceToHost);
       cudaMemcpy(h_arr, d_arr, n * sizeof(int), cudaMemcpyDeviceToHost);
```

```
        temp = h_arr[i];
        h_arr[i] = h_arr[minIdx];
        h_arr[minIdx] = temp;

        cudaMemcpy(d_arr, h_arr, n * sizeof(int), cudaMemcpyHostToDevice);
    }

    cudaMemcpy(h_arr, d_arr, n * sizeof(int), cudaMemcpyDeviceToHost);

    printf("Final Sorted Values:");
    for (i = 0; i < n; i++)
        printf("%d ", h_arr[i]);
    printf("\nAdarsh Ranjan, 230962278");

    cudaFree(d_arr);
    cudaFree(d_minIdx);

    return 0;
}
```

Output:



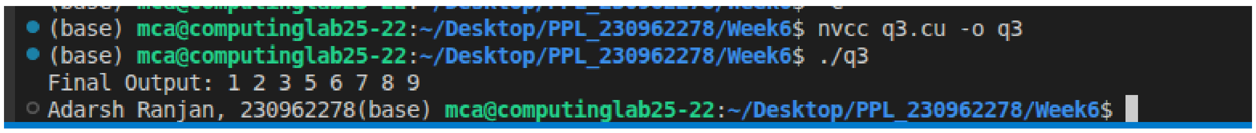
```
● (base) mca@computinglab25-22:~/Desktop/PPL_230962278/Week6$ nvcc q2.cu -o q2
● (base) mca@computinglab25-22:~/Desktop/PPL_230962278/Week6$ ./q2
  Final Sorted Values:2 11 12 22 25 34 64 90
○ Adarsh Ranjan, 230962278(base) mca@computinglab25-22:~/Desktop/PPL_230962278/Week6$
```

Q3:

```
#include <stdio.h>
#include <cuda.h>

__global__ void oddEvenSortStep(int *arr, int n, int phase)
{
   int tid = blockIdx.x * blockDim.x + threadIdx.x;
   int i = 2 * tid + phase;
   if (i + 1 < n)
   {
       if (arr[i] > arr[i + 1])
       {
           int temp = arr[i];
           arr[i] = arr[i + 1];
           arr[i + 1] = temp;
       }
   }
}

int main()
{
   int n = 8;
   int h_arr[8] = {9, 3, 7, 1, 8, 2, 6, 5};
   int *d_arr;
   int i;

   cudaMalloc((void**)&d_arr, n * sizeof(int));
   cudaMemcpy(d_arr, h_arr, n * sizeof(int), cudaMemcpyHostToDevice);

   int threads = n / 2;
   int blockSize = threads;
   int gridSize = 1;

   for (i = 0; i < n; i++)
   {
       int phase = i % 2;
       oddEvenSortStep<<<gridSize, blockSize>>>(d_arr, n, phase);
       cudaDeviceSynchronize();
   }
```

```
    cudaMemcpy(h_arr, d_arr, n * sizeof(int), cudaMemcpyDeviceToHost);


    printf("Final Output: ");
    for (i = 0; i < n; i++)
        printf("%d ", h_arr[i]);
    printf("\nAdarsh Ranjan, 230962278");


    cudaFree(d_arr);

    return 0;
}
```

Output:



```
● (base) mca@computinglab25-22:~/Desktop/PPL_230962278/Week6$ nvcc q3.cu -o q3
● (base) mca@computinglab25-22:~/Desktop/PPL_230962278/Week6$ ./q3
  Final Output: 1 2 3 5 6 7 8 9
○ Adarsh Ranjan, 230962278(base) mca@computinglab25-22:~/Desktop/PPL_230962278/Week6$
```