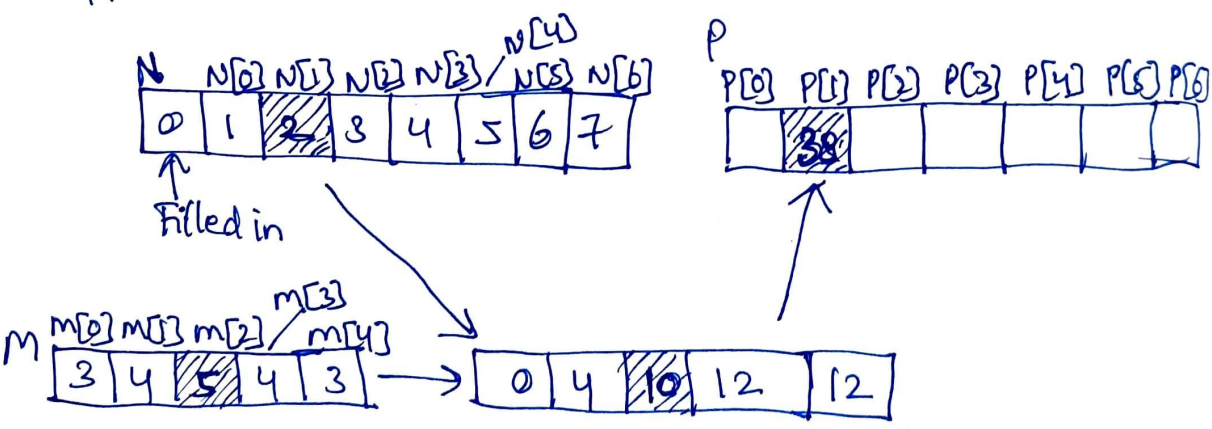
Arunima Singh Thakur, 180905218, Sec C, Roll no. 31, Branch CSE, PCAP Assignment 2,

Ans

1) 1D Sequential Convolution is defined in terms of neighboring elements, so boundary conditions naturally exist for output elements that are close to the ends of an array. For example, when we calculate P[1], there is only one N element to the left of N[1]. That is, there are not enough, N elements to calculate P[1] according to our definition of convolution. A typical approach to handling such a boundary condition is to define a default value to these missing N elements. For most applications, the default value is 0.



$$P[1] = 0 * m[0] + N[0] * m[1] + N[1] * m[2] + N[2] * m[3] + N[3] * m[4]$$
$$= 0 * 3 + 1 * 4 + 2 * 5 + 3 * 4 + 4 * 3$$
$$= 38$$

These missing elements are typically referred to as ghost elements in literature.

2) #include <stdlib.h>
#include <stdio.h>

__global__
void border_prime (int *d_A, int *d_cnt, int N)
{
int Row = blockIdx.y * blockDim.y + threadIdx.y;
int Col = blockIdx.x * blockDim.x + threadIdx.x;

if ((Row == 0) || (Col == 0) || (Row == N-1) || (Col == N-1)) {
if (Row < N && Col < N)
{
int id = Row * N + Col;
int count;
for (int i=1; i<=N; i++)
if(N%i == 0)
count ++;
if(count == 2)
atomicAdd(d_cnt, 1);
}
}
}

int main()
{
int N, count = 0;
scanf ("%d", &N);
~~matt~~ int *arr;
malloc (arr, sizeof (int) * N * N);
for (int i=0; i<N; i++)
for (int j=0; j<N; j++)
scanf ("%d", &(arr[i*N + j]));

int *d_A, *d_cnt;

```
cudaMalloc(&d_A, sizeof(int) * N * N);
cudaMalloc(&d_cnt, sizeof(int));
cudaMemcpy(d_A, arr, sizeof(arr),
    cudaMemcpyHostToDevice);
cudaMemcpy(d_cnt, count, sizeof(int),
    cudaMemcpyHostToDevice);
dim3 dimGrid(2, 2);
dim3 dimBlock(N/2, N/2);
prime <<< dimGrid, dimBlock >>>(d_A,
 d_cnt, N);
 cudaMemcpy(&count, d_cnt, sizeof(int),
     cudaMemcpyDeviceToHost);
 printf("No. of border primes = %d", count);
 cudaFree(d_A);
 cudaFree(d_cnt);
}
```