PPL LAB 6

NAME: ABHINAV SINGH BHAGTANA

ROLL NO: A-029

REG NO: 230905225

Q1.Write a program in CUDA which performs convolution operation on one dimensional input array of size width using a mask array M of size mask_width to produce the resultant one dimensional array P of size width.

#include "cuda_runtime.h"

#include <stdio.h>

__global__ void convolution(int *a,int *m,int *r,int al, int ml) {

```
int idx = blockIdx.x * blockDim.x + threadIdx.x;
if(idx < al)
{
    int start_point = idx - ml / 2;
    int sum=0;
    for(int i=0;i<ml;i++)
    {
        if(start_point + i >= 0 && start_point + i < al)
            sum += a[start_point+i] * m[i];
    }
    r[idx] = sum;
}
```

}

int main() { int width,mask_width; printf("enter length of array\n"); scanf("%d",&width); printf("enter length of mask array\n"); scanf("%d",&mask_width);

```
int array[width],mask[mask_width],result[width];
int *d_array,*d_mask,*d_result;
int size1 = width * sizeof(int);
int size2 = mask_width * sizeof(int);
```

```
printf("enter %d elements:\n",width);
for(int i = 0;i < width;i++)
    scanf("%d",&array[i]);
printf("enter %d elements for mask:\n",mask_width);
for(int i = 0;i < mask_width;i++)
    scanf("%d",&mask[i]);

cudaMalloc((void**)&d_array, size1);
cudaMalloc((void**)&d_mask, size2);
cudaMalloc((void**)&d_result, size1);

cudaMemcpy(d_array,array,size1,cudaMemcpyHostToDevice);
cudaMemcpy(d_mask,mask,size2,cudaMemcpyHostToDevice);

convolution<<<ceil((float)width/256.0),256>>>(d_array,d_mask,d_result,
width,mask_width);

cudaMemcpy(result,d_result,size1,cudaMemcpyDeviceToHost);

printf("result:\n");
for(int i = 0;i < width;i++)
    printf("%d\t",result[i]);
printf("\n");

cudaFree(d_array);
cudaFree(d_mask);
cudaFree(d_result);

return 0;


}
```

OUTPUT:

```
STUDENT@MIT-ICT-LAB5-29:~/230905225/week6$ nvcc q1.cu
STUDENT@MIT-ICT-LAB5-29:~/230905225/week6$ ./a.out
enter length of array
5
enter length of mask array
3
enter 5 elements:
1 2 3 4 5
enter 3 elements for mask:
1 2 3
result:
8       14      20      26      14
STUDENT@MIT-ICT-LAB5-29:~/230905225/week6$
```

Q2.Write a program in CUDA to perform selection sort in parallel.

#include "cuda_runtime.h"

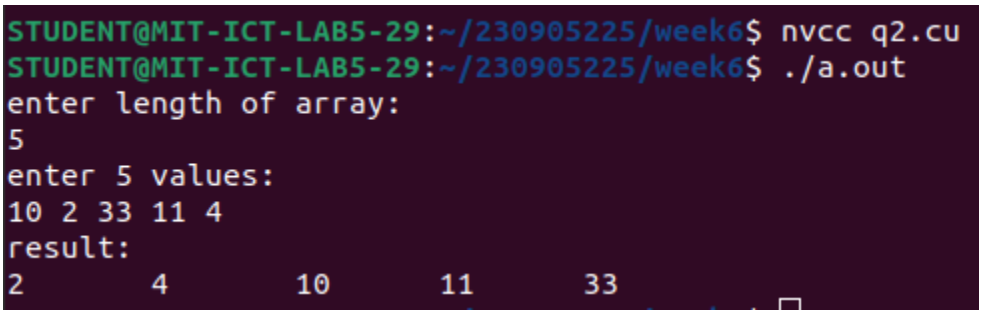
#include <stdio.h>

__**global** __void selSort(int *arr,int len, int *res) { int i = blockIdx.x *blockDim.x + threadIdx.x; if(i<len) { int idx=0; int chosen_element = arr[i]; for(int j = 0;j < len;j++) { if((chosen_element > arr[j]) || (chosen_element == arr[j] && j<i)) idx++; } res[idx] = chosen_element; } } int main() { int n; printf("enter length of array:\n"); scanf("%d",&n);

```
int array[n],result[n];
int *d_array,*d_result;
int size = n * sizeof(int);

printf("enter %d values:\n",n);
for(int i = 0;i < n;i++)
    scanf("%d",&array[i]);

cudaMalloc((void**)&d_array,size);
cudaMalloc((void**)&d_result,size);

cudaMemcpy(d_array,array,size,cudaMemcpyHostToDevice);

selSort<<<ceil((float)n/256.0),256>>>(d_array,n,d_result);

cudaMemcpy(result,d_result,size,cudaMemcpyDeviceToHost);
printf("result:\n");
```

```
for(int i = 0;i < n;i++)
    printf("%d\t",result[i]);
printf("\n");

cudaFree(d_array);
cudaFree(d_result);
return 0;


}
```

OUTPUT:



Q3.write a program in CUDA to perform odd even transposition sort in parallel

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

global void odd(int *a,int len) { int i = blockDim.x * blockIdx.x + threadIdx.x; if(i % 2 != 0 && i < len - 1 && a[i] < a[i+1]) { int temp = a[i]; a[i] = a[i+1]; a[i+1] = temp; } } __global__ void even(int *a,int len) { int i = blockDim.x * blockIdx.x + threadIdx.x; if(i % 2 == 0 && i < len - 1 && a[i] < a[i+1]) { int temp = a[i]; a[i] = a[i+1]; a[i+1] = temp; } } void transpositionSort(int *a,int len) { for(int i = 0;i < (len+1)/2;i++) { odd<<<ceil((float)len/256.0),256>>>(a,len); even<<<ceil((float)len/256.0),256>>>(a,len); } } int main() { int n; printf("enter size of array: "); scanf("%d",&n);

```
int input[n];
int *d_input;
int size = n * sizeof(int);

printf("enter %d elements:\n",n);
for(int i = 0;i < n;i++)
    scanf("%d",&input[i]);
```

```
cudaMalloc((void**)&d_input,size);

cudaMemcpy(d_input,input,size,cudaMemcpyHostToDevice);

transpositionSort(d_input,n);

cudaMemcpy(input,d_input,size,cudaMemcpyDeviceToHost);
printf("result:\n");
for(int i = 0;i < n;i++)
    printf("%d\t",input[i]);
printf("\n");

cudaFree(d_input);
return 0;


}
```

OUTPUT:

```
STUDENT@MIT-ICT-LAB5-29:~/230905225/week6$ nvcc q3.cu
STUDENT@MIT-ICT-LAB5-29:~/230905225/week6$ ./a.out
enter size of array: 10
enter 10 elements:
11 40 1 55 441 2 6 35 74 11
result:
441     74      55      40      35      11      11      6       2       1
```