Name: Manasi. B. Kshirsagar

PRN: S17111009

Roll No.: 06

Class: BE Comp SS

Code:

```
#include<iostream>
#include<cstdio>
#include<cstdlib>
#include<cuda_runtime.h>
using namespace std;


__global__ void minimum(int *input)
{
        int tid=threadIdx.x;
        auto step_size=1;
  int number_of_threads=blockDim.x;

  while(number_of_threads>0)
  {
     if(tid<number_of_threads)
     {
        int first=tid*step_size*2;
        int second=first+step_size;
```

```
        if(input[second]<input[first])
          input[first]=input[second];
      }
      step_size=step_size*2;
      number_of_threads/=2;
  }
}

__global__ void max(int *input)
{
   int tid=threadIdx.x;
   auto step_size=1;
   int number_of_threads=blockDim.x;
   while(number_of_threads>0)
   {
      if(tid<number_of_threads)
      {
         int first=tid*step_size*2;
         int second=first+step_size;
         if(input[second]>input[first])
          input[first]=input[second];
      }
      step_size*=2;
      number_of_threads/=2;
   }
}
```

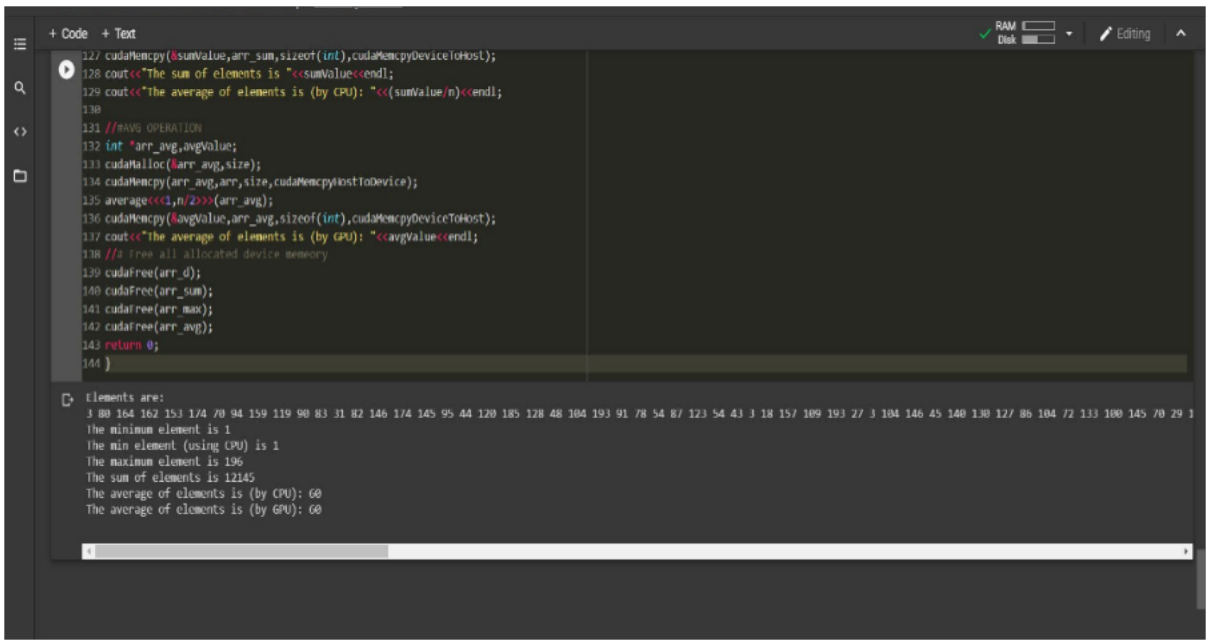
```
__global__ void sum(int *input)
{
    const int tid=threadIdx.x;
    auto  step_size=1;
    int number_of_threads=blockDim.x;
    while(number_of_threads>0)
    {
        if(tid<number_of_threads)
        {
            const int first=tid*step_size*2;
            const int second=first+step_size;
            input[first]=input[first]+input[second];
        }
    step_size = step_size*2;;
                number_of_threads =number_of_threads/2;
    }
}


__global__ void average(int *input) //You can use above sum() to calculate
sum and divide it by num_of_elememts
{
    const int tid=threadIdx.x;
    auto  step_size=1;
    int number_of_threads=blockDim.x;
    int totalElements=number_of_threads*2;
    while(number_of_threads>0)
```

```
    {
        if(tid<number_of_threads)
        {
            const int first=tid*step_size*2;
            const int second=first+step_size;
            input[first]=input[first]+input[second];
        }
        step_size = step_size*2;;
                number_of_threads =number_of_threads/2;
    }
    input[0]=input[0]/totalElements;
}

int main()
{
int n;
n=200;
srand(n);
int *arr=new int[n];
int min=20000; //Any Large Number would work
 cout<<"Elements are: "<<endl;
   //# Generate Input array using rand()
         for(int i=0;i<n;i++)
         {
         arr[i]=rand()%n;
      if(arr[i]<min)
```

```
        min=arr[i];
    cout<<arr[i]<<" ";
        }
        int size=n*sizeof(int); //calculate no. of bytes for array
        int *arr_d,result1;

//# Allocate memory for min Operation
cudaMalloc(&arr_d,size);
cudaMemcpy(arr_d,arr,size,cudaMemcpyHostToDevice);
minimum<<<1,n/2>>>(arr_d);
cudaMemcpy(&result1,arr_d,sizeof(int),cudaMemcpyDeviceToHost);
cout<<endl<<"The minimum element is "<<result1<<endl;
cout<<"The min element (using CPU) is "<<min<<endl;


//#MAX OPERATION
int *arr_max,maxValue;
cudaMalloc(&arr_max,size);
cudaMemcpy(arr_max,arr,size,cudaMemcpyHostToDevice);
max<<<1,n/2>>>(arr_max);
cudaMemcpy(&maxValue,arr_max,sizeof(int),cudaMemcpyDeviceToHost);
cout<<"The maximum element is "<<maxValue<<endl;

//SUM OPERATION
int *arr_sum,sumValue;
cudaMalloc(&arr_sum,size);
```

```
cudaMemcpy(arr_sum,arr,size,cudaMemcpyHostToDevice);
sum<<<1,n/2>>>(arr_sum);
cudaMemcpy(&sumValue,arr_sum,sizeof(int),cudaMemcpyDeviceToHost);
cout<<"The sum of elements is "<<sumValue<<endl;
cout<<"The average of elements is (by CPU): "<<(sumValue/n)<<endl;

//#AVG OPERATION
int *arr_avg,avgValue;
cudaMalloc(&arr_avg,size);
cudaMemcpy(arr_avg,arr,size,cudaMemcpyHostToDevice);
average<<<1,n/2>>>(arr_avg);
cudaMemcpy(&avgValue,arr_avg,sizeof(int),cudaMemcpyDeviceToHost);
cout<<"The average of elements is (by GPU): "<<avgValue<<endl;
//# Free all allocated device memeory
cudaFree(arr_d);
cudaFree(arr_sum);
cudaFree(arr_max);
cudaFree(arr_avg);
return 0;
}
```

Output:


+ Code + Text
RAM
Disk
Editing
cudaMemcpy(&sumValue,arr_sum,sizeof(int),cudaMemcpyDeviceToHost);
cout<<"The sum of elements is "<<sumValue<<endl;
cout<<"The average of elements is (by CPU): "<<(sumValue/n)<<endl;
//#AVG OPERATION
int *arr_avg,avgValue;
cudaMalloc(&arr_avg,size);
cudaMemcpy(arr_avg,arr,size,cudaMemcpyHostToDevice);
average<<<1,n/2>>>(arr_avg);
cudaMemcpy(&avgValue,arr_avg,sizeof(int),cudaMemcpyDeviceToHost);
cout<<"The average of elements is (by GPU): "<<avgValue<<endl;
//# free all allocated device memeory
cudaFree(arr_d);
cudaFree(arr_sum);
cudaFree(arr_max);
cudaFree(arr_avg);
return 0;
}
Elements are:
3 80 164 162 153 174 78 94 159 119 90 83 31 82 146 174 145 95 44 120 185 128 48 104 193 91 78 54 87 123 54 43 3 18 157 109 193 27 3 104 146 45 140 130 127 86 104 72 133 100 145 70 29 1
The minimum element is 1
The min element (using CPU) is 1
The maximum element is 196
The sum of elements is 12145
The average of elements is (by CPU): 60
The average of elements is (by GPU): 60