Name: Prem Vinod Bansod
Roll no. 41310

Assignment 1

```
%%cu
#include<bits/stdc++.h>
#include<stdio.h>
#define n 8
using namespace std;

__global__ void minimum(float* arr)
{
   int t = threadIdx.x;
   int step = 1;
   int nt = blockDim.x;
   while(nt>0)
   {
      int fi = t*step*2;
      int si = fi + step;
      if(fi<n&&si<n)
      {
         printf("%d\t%d\n",fi,si);
         if(arr[fi]>arr[si])
         {
           arr[fi] = arr[si];
         }
      }
      __syncthreads();
      step = step<<1;
      nt = nt>>1;
   }
}

__global__ void maximum(float* arr)
{
   int t = threadIdx.x;
   int step = 1;
   int nt = blockDim.x;
   while(nt>0)
   {
      int fi = t*step*2;
      int si = fi + step;
      if(fi<n&&si<n)
```

```
        {
            if(arr[fi]<arr[si])
            {
                arr[fi] = arr[si];
            }
        }
        __syncthreads();
        step = step<<1;
        nt = nt>>1;
    }
}

__global__ void sum(float* arr)
{
    int t = threadIdx.x;
    int step = 1;
    int nt = blockDim.x;
    while(nt>0)
    {
        int fi = t*step*2;
        int si = fi + step;
        if(fi<n&&si<n)
        {
            arr[fi] = arr[fi] + arr[si];
        }
        __syncthreads();
        step = step<<1;
        nt = nt>>1;
    }
}

__global__ void submean(float* arr,float mean)
{
    int t = threadIdx.x;
    arr[t] = pow(arr[t] - mean,2);
}


void randnumgen(float *arr)
{
  for(int i=0;i<n;i++)
  {
    arr[i] = rand()%100;
    cout<<arr[i]<<" ";
  }
```

```
  cout<<endl;
}


int main()
{
    float arr[n];

    randnumgen(arr);
    float* arr_p;
    float res;
    int size = n * sizeof(float);

    cudaMalloc((void **)&arr_p,size);

    cudaMemcpy(arr_p,arr,size,cudaMemcpyHostToDevice);

    int threads = n/2;

    minimum<<<1,threads>>>(arr_p);

    cudaMemcpy(&res,arr_p,sizeof(float),cudaMemcpyDeviceToHost);

    cout<<"Minimum Element = "<<res<<"\n";

    cudaMemcpy(arr_p,arr,size,cudaMemcpyHostToDevice);

    maximum<<<1,threads>>>(arr_p);

    cudaMemcpy(&res,arr_p,sizeof(float),cudaMemcpyDeviceToHost);

    cout<<"Maximum Element = "<<res<<"\n";

    cudaMemcpy(arr_p,arr,size,cudaMemcpyHostToDevice);

    sum<<<1,threads>>>(arr_p);

    cudaMemcpy(&res,arr_p,sizeof(float),cudaMemcpyDeviceToHost);

    float mean = res/n;

    cout<<"Arithmatic mean  = "<<mean<<"\n";

    cudaMemcpy(arr_p,arr,size,cudaMemcpyHostToDevice);
```

```
    submean<<<1,n>>>(arr_p,mean);

    sum<<<1,threads>>>(arr_p);

    cudaMemcpy(&res,arr_p,sizeof(float),cudaMemcpyDeviceToHost);

    cout<<"Variance = "<<(res/n)<<"\n";

    cout<<"Standard Deviation = "<<sqrt(res/n)<<"\n";

    return 0;
}
```

output: