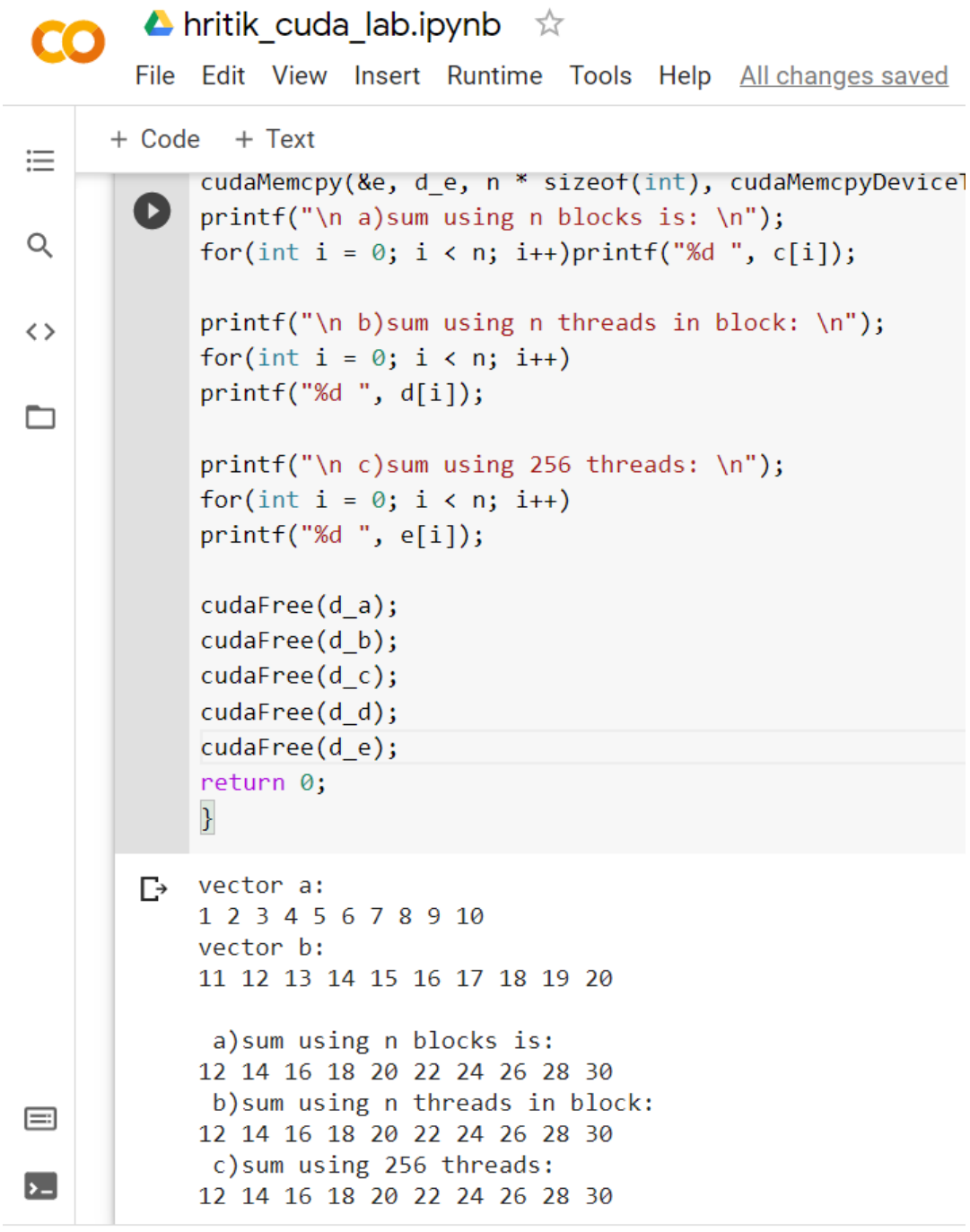
HRITIK AJAY BANSAL
CSE A 15
180905105
PP LAB WEEK 5

Q1) %%cu

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
__global__ void sumNBlocks(int *a, int *b, int *c, int n) {
int i = blockIdx.x;
c[i] = a[i] + b[i];
}
__global__ void sumNThreads(int *a, int *b, int *c, int n) {
int i = threadIdx.x;
c[i] = a[i] + b[i];
}
__global__ void sum256Threads(int *a, int *b, int *c, int n) {
int i = threadIdx.x + blockIdx.x * blockDim.x;if (i < n)
c[i] = a[i] + b[i];
}
int main() {
int *d_a, *d_b, *d_c, *d_d, *d_e;
int n = 10;
int a[n] = {1,2,3,4,5,6,7,8,9,10};
int b[n] = {11,12,13,14,15,16,17,18,19,20};
int c[n], d[n], e[n];
printf("vector a: \n");
for(int i = 0; i < n; i++)
printf("%d ", a[i]);
printf("\n");
printf("vector b: \n");
for(int i = 0; i < n; i++)
printf("%d ", b[i]);
printf("\n");
cudaMalloc((void **)&d_a, n * sizeof(int));
cudaMalloc((void **)&d_b, n * sizeof(int));
cudaMalloc((void **)&d_c, n * sizeof(int));
cudaMalloc((void **)&d_d, n * sizeof(int));
cudaMalloc((void **)&d_e, n * sizeof(int));
cudaMemcpy(d_a, &a, n * sizeof(int), cudaMemcpyHostToDevice);
cudaMemcpy(d_b, &b, n * sizeof(int), cudaMemcpyHostToDevice);

//#launch kernels
sumNBlocks<<<n,1>>>(d_a, d_b, d_c, n);
sumNThreads<<<1,n>>>(d_a, d_b, d_d, n);
sum256Threads<<<ceil(n / 256.0), 256>>>(d_a, d_b, d_e, n);

cudaMemcpy(&c, d_c, n * sizeof(int), cudaMemcpyDeviceToHost);
cudaMemcpy(&d, d_d, n * sizeof(int), cudaMemcpyDeviceToHost);
cudaMemcpy(&e, d_e, n * sizeof(int), cudaMemcpyDeviceToHost);
printf("\n a)sum using n blocks is: \n");
for(int i = 0; i < n; i++)printf("%d ", c[i]);
```

```
printf("\n b)sum using n threads in block: \n");
for(int i = 0; i < n; i++)
printf("%d ", d[i]);

printf("\n c)sum using 256 threads: \n");
for(int i = 0; i < n; i++)
printf("%d ", e[i]);

cudaFree(d_a);
cudaFree(d_b);
cudaFree(d_c);
cudaFree(d_d);
cudaFree(d_e);
return 0;
}
```

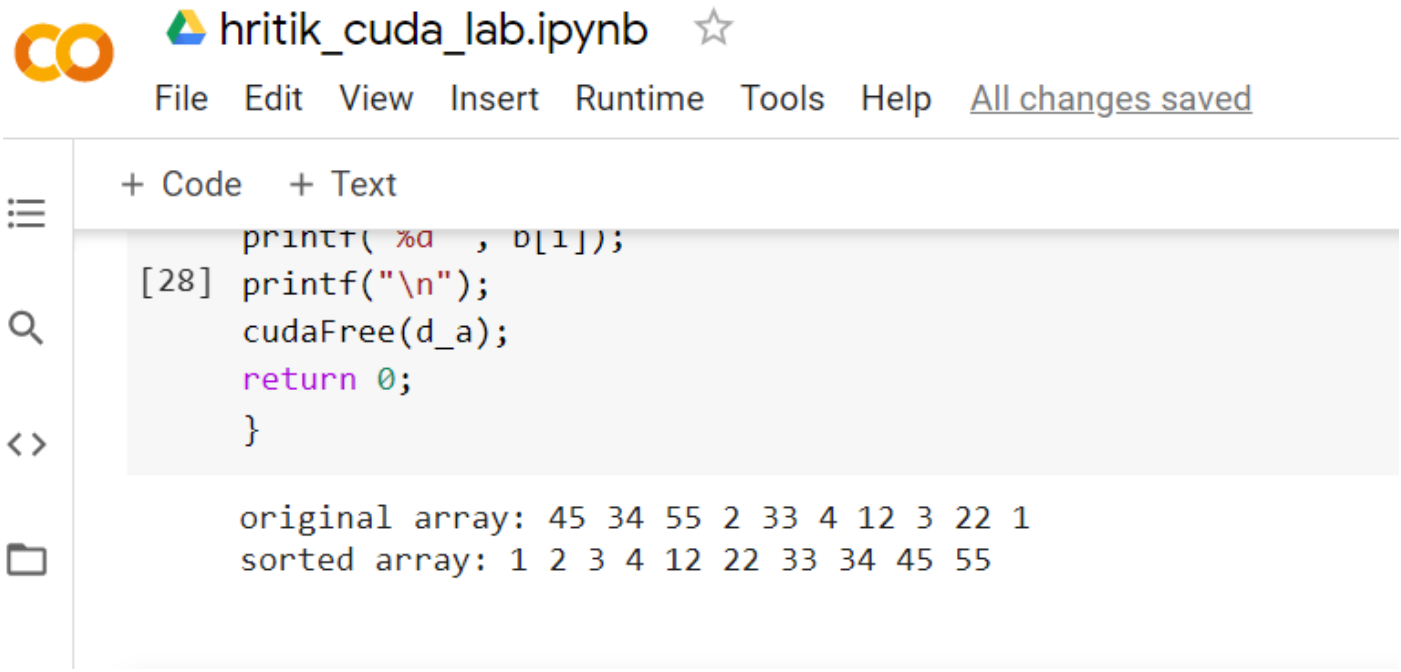
Output:



Q2) %%cu

```
#include <stdio.h>
#include <stdlib.h>
__global__ void parallelSelectionSort(int *a, int n) {
int i = threadIdx.x;
int data = a[i];
int pos = 0;
for (int j = 0; j < n; j++) {
if (a[j] < data || (a[j] == data && j < i))
pos++;
}
a[pos] = data;
}
int main() {
int *d_a;
int n = 10;
int a[n] = {45,34,55,2,33,4,12,3,22,1};
int b[n]; //output array
printf("original array: ");
for(int i = 0; i < n; i++)
printf("%d ", a[i]);
printf("\n");
cudaMalloc((void **)&d_a, n * sizeof(int));
cudaMemcpy(d_a, &a, n * sizeof(int), cudaMemcpyHostToDevice);

//#LAUNCH KERNEL
parallelSelectionSort<<<1,n>>>(d_a, n);
cudaError err = cudaMemcpy(&b, d_a, n * sizeof(int),
cudaMemcpyDeviceToHost);if(err!=cudaSuccess) {
printf("CUDA error copying to Host: %s\n",
cudaGetErrorString(err));
}
printf("sorted array: ");
for(int i = 0; i < n; i++)
printf("%d ", b[i]);
printf("\n");
cudaFree(d_a);
return 0;
}
```

Output:



Q3) %%cu

```
#include <stdio.h>
#include <stdlib.h>
__global__ void oddEvenTransposSort(int *a, int n) {
int i = threadIdx.x;
if ((i % 2 == 1) && i < n - 1) {
if (a[i] > a[i+1]) {
int temp = a[i];
a[i] = a[i+1];
a[i+1] = temp;
}
}
}
__global__ void evenOddTransposSort(int *a, int n) {
int i = threadIdx.x;
if ((i % 2 == 0) && i < n - 1) {
if (a[i] > a[i+1]) {
int temp = a[i];
a[i] = a[i+1];
a[i+1] = temp;
}
}
}
int main() {
int *d_a;
int n = 9;
int a[n] = {12,44,5,-1,0,111,3,-7,66};
int b[n];printf("original array: ");
for(int i = 0; i < n; i++)
printf("%d ", a[i]);
printf("\n");
cudaMalloc((void **)&d_a, n * sizeof(int));
cudaMemcpy(d_a, &a, n * sizeof(int), cudaMemcpyHostToDevice);
for (int i = 0; i <= n / 2; i++) {
    //#kernel launch
```

```
oddEvenTransposSort<<<1,n>>>(d_a, n);
evenOddTransposSort<<<1,n>>>(d_a, n);
}
cudaError err = cudaMemcpy(&b, d_a, n * sizeof(int),
cudaMemcpyDeviceToHost);
if(err!=cudaSuccess) {
printf("CUDA error copying to Host: %s\n",
cudaGetErrorString(err));
}
printf("sorted array: ");
for(int i = 0; i < n; i++)
printf("%d ", b[i]);
printf("\n");
cudaFree(d_a);
return 0;
}
```

Output: