मौलाना आज़ाद राष्ट्रीय प्रौद्योगिकी संस्थान, भोपाल – 462003
MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY, BHOPAL – 462003

कंप्यूटर विज्ञान एवं अभियांत्रिकी विभाग
(Department of Computer Science and Engineering)

लैब मैन्युअल (LAB Manual)

सीएसई– तृतीय वर्ष / छठवां सेमेस्टर (CSE - III Year / VI Semester)

सत्र: जनवरी- जून, २०२५(Session: Jan - June, 2025)

हडूप कूड़ा लैब (सीएसई - 326) / HADOOP CUDA LAB (CSE - 326)

LAB: 01

1. गूगल को लैब पर कूड़ा एनवायरनमेंट कॉन्फ़िगर करे।

Configure CUDA environment on GOOGLE COLAB.

- ✓ Log In to Google CoLab
- ✓ Open a new JupyterNoteBook
- ✓ Change the Run-Time to TPU and SAVE
- ✓ Check for CUDA Version: **!nvcc–version**

  Output:

  nvcc: NVIDIA (R) Cuda compiler driver
  Copyright (c) 2005-2024 NVIDIA Corporation
  Built on Thu_Jun__6_02:18:23_PDT_2024
  Cuda compilation tools, release 12.5, V12.5.82
  Build cuda_12.5.r12.5/compiler.34385749_0
- ✓ Install Jupyter for CUDA: **!pip install nvcc4jupyter**

  Output:

  Collecting nvcc4jupyter
  Downloading nvcc4jupyter-1.2.1-py3-none-any.whl.metadata (5.1 kB)
  Downloading nvcc4jupyter-1.2.1-py3-none-any.whl (10 kB)
  Installing collected packages: nvcc4jupyter
  Successfully installed nvcc4jupyter-1.2.1
- ✓ Load the Jupyter: **%load_ext nvcc4jupyter**

  Output:

  Detected platform "Colab". Running its setup...
  Source files will be saved in "/tmp/tmp24ieq317".
- ✓ Implement the programs in CUDA C and RUN

---

2. डिवाइस की जानकारी, ड्राइवर संस्करण, रनटाइम संस्करण, गणना क्षमता, एवं ग्लोबल मेमोरी के बाइट्स की जानकारी प्रदर्शित करने के लिए कूड़ा सी में प्रोग्राम लिखे।

Write a program in CUDA C to check and display Device Information including Driver Version, Runtime Version, Compute Capability, and Bytes of Global Memory.

Write and execute the Code below in a CUDA Cell:

```
%%writefile device_info.cu
#include <stdio.h>
#include <cuda_runtime.h>

int main() {
   int deviceCount = 0;              cudaError_terror_id = cudaGetDeviceCount(&deviceCount);
   if (error_id != cudaSuccess) {
printf("cudaGetDeviceCount returned %d\n-> %s\n", (int)error_id, cudaGetErrorString(error_id));
      return 1;              }

printf("Detected %d CUDA Capable device(s)\n", deviceCount);

   for (int dev = 0; dev <deviceCount; ++dev) {              cudaDevicePropdeviceProp;
cudaGetDeviceProperties(&deviceProp, dev);

printf("\nDevice %d: \"%s\"\n", dev, deviceProp.name);
printf("  CUDA Capability: %d.%d\n", deviceProp.major, deviceProp.minor);
printf("  Total Global Memory: %lu MB\n", (unsigned long)(deviceProp.totalGlobalMem / 1024 / 1024));
printf("  Multiprocessors: %d\n", deviceProp.multiProcessorCount);
printf("  Max Threads Per Block: %d\n", deviceProp.maxThreadsPerBlock);                    }

   return 0;                  }
```

Write and execute the Code below in another CUDA Cell:

```
!nvcc device_info.cu -o device_info
!./device_info
```

Output:

```
Detected 1 CUDA Capable device(s)

Device 0: "Tesla T4"              CUDA Capability: 7.5              Total Global Memory: 15095 MB
  Multiprocessors: 40             Max Threads Per Block: 1024
```

---

3. सीपीयू और जीपीयू से टर्मिनल पर "Hello World" प्रदर्शित करने के लिए कूड़ा सी में प्रोग्राम लिखे।

Write a program in CUDA C to display “Hello World” on terminal from CPU and GPU.

```
%%cuda
#include<stdio.h>

int main() {
printf("Hello World \n");
   return 0;                }
```

Output:

Hello World

---

4. सीपीयू और जीपीयू से टर्मिनल पर अपनानाम *n* बार प्रदर्शित करने के लिए कूड़ा सी में प्रोग्राम लिखे।

Write a program in CUDA C to display your name *n* times on terminal from CPU and GPU.

```
%%cuda
#include<stdio.h>
#define N 20

int main() {
for(int i = 1; i<= N; i++)
printf("My Name is Michael Jackson ! \n");
   return 0;                }
```

Output:

My Name is Michael Jackson !
My Name is Michael Jackson !
My Name is Michael Jackson !
My Name is Michael Jackson !
My Name is Michael Jackson !

Printed 20 Times in separate lines

मौलाना आज़ाद राष्ट्रीय प्रौद्योगिकी संस्थान, भोपाल – 462003
MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY, BHOPAL – 462003

कंप्यूटर विज्ञान एवं अभियांत्रिकी विभाग
(Department of Computer Science and Engineering)

LAB: 02

1. दो वेक्टरों को जोड़ने के लिए कूड़ा सी में यूजर ड्रिवन प्रोग्राम इम्प्लीमेंट करें। इनपुट के रूप में वेक्टर का साइज (एलिमेंट्सकीसंख्या) लें। सीपीयू पर समान कार्य के लिए आवश्यक समय की तुलना करें।

Implement a user-driven program in CUDA C for addition of two vectors. Take size of vector (number of elements) as input. Compare the time required for the same task on CPU.

```
%%shell
nvcc -arch=sm_70 -o vector_add -x cu - <<'EOF'
#include <stdio.h>
#define N 100

__global__ void vectorAdd(float *a, float *b, float *c, int n) {
   // Only one thread does the work
   for (int i = 0; i< n; i++) {
      c[i] = a[i] + b[i];              }                    }

int main() {
size_t size = N * sizeof(float);
   float h_a[N], h_b[N], h_c[N];

   for (int i = 0; i< N; i++) {
h_a[i] = i;                    h_b[i] = i * 2;              }

   float *d_a, *d_b, *d_c;
cudaMalloc((void**)&d_a, size);
cudaMalloc((void**)&d_b, size);
cudaMalloc((void**)&d_c, size);

cudaMemcpy(d_a, h_a, size, cudaMemcpyHostToDevice);
cudaMemcpy(d_b, h_b, size, cudaMemcpyHostToDevice);

vectorAdd<<<1, 1>>>(d_a, d_b, d_c, N);
cudaDeviceSynchronize();
cudaMemcpy(h_c, d_c, size, cudaMemcpyDeviceToHost);

   for (int i = 0; i< N; i++) {                printf("h_c[%d] = %f\n", i, h_c[i]);                }
```

```
cudaFree(d_a);          cudaFree(d_b);          cudaFree(d_c);

   return 0;            }
EOF

# Run the compiled binary
./vector_add
```

Output:

```
h_c[0] = 20.000000      h_c[1] = 30.000000      h_c[2] = 40.000000
h_c[3] = 50.000000      h_c[4] = 60.000000      h_c[5] = 70.000000
h_c[6] = 30.000000      h_c[7] = 40.000000      h_c[8] = 50.000000
h_c[9] = 60.000000      Sum of Elements is : 450.000000
```

---

2. वेक्टर के सभी एलिमेंट्स के योग की गणना के लिए कूड़ा सी में यूजर ड्रिवन प्रोग्राम इम्प्लीमेंट करें। वेक्टर के मैक्स और मिन एलिमेंट को भी ज्ञात करें। सीपीयू पर समान कार्य के लिए आवश्यक समय की तुलना करें।

Implement a user-driven program in CUDA C for computing sum of all elements of a vector. Also determine MAX and MIN element from the vector. Compare the time required for the same task on CPU.

```
%%shell
nvcc -arch=sm_70 -lcudart -o max_min_single_thread -x cu - <<'EOF'

#include <stdio.h>
#define N 10

__global__ void findMaxMinSum(float *arr, float *maxVal, float *minVal, float *tot) {
   if (threadIdx.x == 0) {
      *maxVal = arr[0];              *minVal = arr[0];

      for (int i = 1; i< N; i++) {   *tot = *tot + arr[i];

         if (arr[i] > *maxVal) {     *maxVal = arr[i];          }
         if (arr[i] < *minVal) {     *minVal = arr[i];          }
      }          }              }

int main() {
   float h_arr[N];
for(int i = 0; i< N; i++)          {
h_arr[i] = (3 * (i + 1) + 7) % 5;          }
```

```
for(int i = 0; i< N; i++)                    {
printf("h_arr[%d] = %f \n", i, h_arr[i]);                    }

   float *d_arr, *d_max, *d_min, *d_tot;
   float h_max = h_arr[0];              float h_min = h_arr[0];                    float h_tot = 0;

cudaMalloc((void**)&d_arr, N * sizeof(float));
cudaMalloc((void**)&d_max, sizeof(float));
cudaMalloc((void**)&d_min, sizeof(float));
cudaMalloc((void**)&d_tot, sizeof(float));

cudaMemcpy(d_arr, h_arr, N * sizeof(float), cudaMemcpyHostToDevice);

findMaxMinSum<<<1, 1>>>(d_arr, d_max, d_min, d_tot);
cudaDeviceSynchronize();

cudaMemcpy(&h_max, d_max, sizeof(float), cudaMemcpyDeviceToHost);
cudaMemcpy(&h_min, d_min, sizeof(float), cudaMemcpyDeviceToHost);
cudaMemcpy(&h_tot, d_tot, sizeof(float), cudaMemcpyDeviceToHost);

printf("Max Value: %f\n", h_max);
printf("Min Value: %f\n", h_min);
printf("Sum of Elements : %f \n", h_tot);

cudaFree(d_arr);              cudaFree(d_max);              cudaFree(d_min);

   return 0;              }
EOF

# Run the compiled CUDA program
./max_min_single_thread
```

Output:

```
h_arr[0] = 0.000000          h_arr[1] = 3.000000          h_arr[2] = 1.000000
h_arr[3] = 4.000000          h_arr[4] = 2.000000          h_arr[5] = 0.000000
h_arr[6] = 3.000000          h_arr[7] = 1.000000          h_arr[8] = 4.000000
h_arr[9] = 2.000000
Max Value: 4.000000          Min Value: 0.000000          Sum of Elements : 20.000000
```

मौलाना आज़ाद राष्ट्रीय प्रौद्योगिकी संस्थान, भोपाल – 462003
MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY, BHOPAL – 462003

कंप्यूटर विज्ञान एवं अभियांत्रिकी विभाग
(Department of Computer Science and Engineering)

LAB: 03

1. मैट्रिक्स एडिशन के लिए कूड़ा सी में यूजर ड्रिवन प्रोग्राम इम्प्लीमेंट करें। इनपुट के रूप में मैट्रिक्स का आकार (rows एवं columns की संख्या) लें।

Implement a user-driven program in CUDA C for matrix addition. Take size of matrix (number of rows and columns) as input.

```
%%shell
nvcc -arch=sm_70 -lcudart -o matrix_add_single_thread -x cu - <<'EOF'

#include <stdio.h>
#define Rows 3
#define Columns 3

__global__ void matrixAdd(float *A, float *B, float *C) {
   if (threadIdx.x == 0) {
      for (int i = 0; i< Rows; i++) {
         for (int j = 0; j < Columns; j++) {
            int index = i * Rows + j;
            C[index] = A[index] + B[index];              }
      }           }                 }

int main() {
   float h_A[Rows][Columns], h_B[Rows][Columns], h_C[Rows][Columns];
for(int i = 0; i< Rows; i++) {
for(int j = 0; j < Columns; j++) {
h_A[i][j] = 3;            h_B[i][j] = 5;            }                 }

   float *d_A, *d_B, *d_C;

cudaMalloc((void**)&d_A, Rows * Columns * sizeof(float));
cudaMalloc((void**)&d_B, Rows * Columns * sizeof(float));
cudaMalloc((void**)&d_C, Rows * Columns * sizeof(float));

cudaMemcpy(d_A, h_A, Rows * Columns * sizeof(float), cudaMemcpyHostToDevice);
```

```
cudaMemcpy(d_B, h_B, Rows * Columns * sizeof(float), cudaMemcpyHostToDevice);

matrixAdd<<<1, 1>>>(d_A, d_B, d_C);
cudaDeviceSynchronize();

cudaMemcpy(h_C, d_C, Rows * Columns * sizeof(float), cudaMemcpyDeviceToHost);

printf("Matrix C (A + B):\n");
   for (int i = 0; i< Rows; i++) {
      for (int j = 0; j < Columns; j++) {
printf("%f ", h_C[i][j]);                  }
printf("\n");              }

cudaFree(d_A);                cudaFree(d_B);                cudaFree(d_C);

   return 0;              }
EOF

# Run the compiled program
./matrix_add_single_thread
```

Output:

```
Matrix C (A + B):
8.000000        8.000000        8.000000
8.000000        8.000000        8.000000
8.000000        8.000000        8.000000
```

---

2. मैट्रिक्स मल्टिप्लिकेशन के लिए कूड़ा सी में यूजर ड्रिवन प्रोग्राम इम्प्लीमेंट करें। इनपुट के रूप में मैट्रिक्स का आकार (rows एवं columns की संख्या) लें।

Implement a user-driven program in CUDA C for matrix addition. Take size of matrix (number of rows and columns) as input.

```
%%shell
nvcc -arch=sm_70 -lcudart -o matrix_multiply_single_thread -x cu - <<'EOF'

#include <stdio.h>
#define N 5
```

```
__global__ void matrixMultiply(float *A, float *B, float *C) {
   if (threadIdx.x == 0) {
      for (int i = 0; i< N; i++) {
         for (int j = 0; j < N; j++) {
C[i * N + j] = 0;

            for (int k = 0; k < N; k++) {
C[i * N + j] += A[i * N + k] * B[k * N + j];            }
         }                  }                  }                  }

int main() {
   float h_A[N][N], h_B[N][N], h_C[N][N];

for(int i = 0; i< N; i++) {
for(int j = 0; j < N; j++) {
h_A[i][j] = 5;            h_B[i][j] = 7;
    }             }

   float *d_A, *d_B, *d_C;

cudaMalloc((void**)&d_A, N * N * sizeof(float));
cudaMalloc((void**)&d_B, N * N * sizeof(float));
cudaMalloc((void**)&d_C, N * N * sizeof(float));

cudaMemcpy(d_A, h_A, N * N * sizeof(float), cudaMemcpyHostToDevice);
cudaMemcpy(d_B, h_B, N * N * sizeof(float), cudaMemcpyHostToDevice);

matrixMultiply<<<1, 1>>>(d_A, d_B, d_C);
cudaDeviceSynchronize();

cudaMemcpy(h_C, d_C, N * N * sizeof(float), cudaMemcpyDeviceToHost);

printf("Matrix C (A * B):\n");
   for (int i = 0; i< N; i++) {
      for (int j = 0; j < N; j++) {
printf("%f ", h_C[i][j]);                  }
printf("\n");              }
```

```
cudaFree(d_A);          cudaFree(d_B);          cudaFree(d_C);

   return 0;          }
EOF

# Run the compiled program
./matrix_multiply_single_thread
```

Output:

```
Matrix C (A * B):
175.000000  175.000000  175.000000  175.000000  175.000000
175.000000  175.000000  175.000000  175.000000  175.000000
175.000000  175.000000  175.000000  175.000000  175.000000
175.000000  175.000000  175.000000  175.000000  175.000000
175.000000  175.000000  175.000000  175.000000  175.000000
```

मौलाना आज़ाद राष्ट्रीय प्रौद्योगिकी संस्थान, भोपाल – 462003
MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY, BHOPAL – 462003

कंप्यूटर विज्ञान एवं अभियांत्रिकी विभाग
(Department of Computer Science and Engineering)

LAB: 04

1. दो वेक्टरों को जोड़ने के लिए कूड़ा सी में थ्रेड्स का उपयोग कर के यूजर ड्रिवन प्रोग्राम इम्प्लीमेंट करें। इनपुट के रूप में वेक्टर का साइज (एलिमेंट्स की संख्या) लें। सीपीयू पर समान कार्य के लिएआवश्यक समय की तुलना करें।

Implement a user-driven program in CUDA C for addition of two vectors using the concept of threads. Take size of vector (number of elements) as input. Compare the time required for the same task on CPU.

```
%%shell
nvcc -arch=sm_70 -o vector_add -x cu - <<'EOF'

#include <stdio.h>
#include<iostream>
#include<cuda_runtime.h>
#define N 1000

__global__ void vectorAdd(float *a, float *b, float *c) {
  int idx = threadIdx.x + blockIdx.x * blockDim.x;
       if (idx< N) {                  c[idx] = a[idx] + b[idx];                  }                  }

int main() {
size_t size = N * sizeof(float);
   float h_a[N], h_b[N], h_c[N];

   for (int i = 0; i< N; i++) {
h_a[i] = 10;                h_b[i] = 20;                }

   float *d_a, *d_b, *d_c;
cudaMalloc((void**)&d_a, size);
cudaMalloc((void**)&d_b, size);
cudaMalloc((void**)&d_c, size);

cudaMemcpy(d_a, h_a, size, cudaMemcpyHostToDevice);
cudaMemcpy(d_b, h_b, size, cudaMemcpyHostToDevice);

   int tpb = 5;              int bpt = (N + tpb - 1) / tpb;
```

```
vectorAdd<<<bpt, tpb>>>(d_a, d_b, d_c);
cudaDeviceSynchronize();
cudaError_t err = cudaGetLastError();
   if (err != cudaSuccess) {
std::cerr<< "CUDA Kernel launch failed: " <<cudaGetErrorString(err) <<std::endl;
   }

cudaMemcpy(h_c, d_c, size, cudaMemcpyDeviceToHost);

   for (int i = 0; i< N; i++) {
printf("h_c[%d] = %f\n", i, h_c[i]);              }

cudaFree(d_a);                  cudaFree(d_b);                  cudaFree(d_c);

   return 0;              }
EOF

# Run the compiled binary
./vector_add
```

Output:

```
h_c[0] = 30.000000
h_c[1] = 30.000000
h_c[2] = 30.000000
h_c[3] = 30.000000
h_c[4] = 30.000000
------------------------
Printed 1000 Times
```

---

2 वेक्टर के सभी तत्वों के योग की गणना के लिए कूड़ा सी में थ्रेड्स का उपयोग कर के यूजर ड्रिवन प्रोग्राम इम्प्लीमेंट करें। वेक्टर के मैक्स और मिन तत्व को भी ज्ञात करें। सीपीयू पर समान कार्य के लिए आवश्यक समय की तुलना करें।

Implement a user-driven program in CUDA C for computing sum of all elements of a vector using the concept of threads. Also determine MAX and MIN element from the vector. Compare the time required for the same task on CPU.

```
%%shell
nvcc -arch=sm_70 -o vector_minmaxsum -x cu - <<'EOF'

#include <iostream>
#include <cuda_runtime.h>
#include <float.h>

#define N 25
#define THREADS_PER_BLOCK 8

__global__ void computeSumMinMax(const float *input, float *sumOut, float *minOut, float
*maxOut) {
   __shared__ float sumShared[THREADS_PER_BLOCK];
   __shared__ float minShared[THREADS_PER_BLOCK];
   __shared__ float maxShared[THREADS_PER_BLOCK];

   int tid = threadIdx.x;                    int i = blockIdx.x * blockDim.x + tid;

   float val = (i< N) ? input[i] : 0.0f;
sumShared[tid] = val;
minShared[tid] = (i< N) ?val : FLT_MAX;
maxShared[tid] = (i< N) ?val : -FLT_MAX;

   __syncthreads();

   for (int stride = blockDim.x / 2; stride > 0; stride >>= 1) {
      if (tid< stride) {
sumShared[tid] += sumShared[tid + stride];
minShared[tid] = fminf(minShared[tid], minShared[tid + stride]);
maxShared[tid] = fmaxf(maxShared[tid], maxShared[tid + stride]);
      }
      __syncthreads();
   }

   if (tid == 0) {
sumOut[blockIdx.x] = sumShared[0];                    minOut[blockIdx.x] = minShared[0];
maxOut[blockIdx.x] = maxShared[0];                    }                    }
```

```
int main() {
   float *h_data = new float[N];
   for (int i = 0; i< N; ++i) {
h_data[i] = static_cast<float>(rand() % 1000); // Random data between 0 and 1000
   }

std::cout<< "Initial data sample: ";
   for (int i = 0; i< N; ++i) {              std::cout<<h_data[i] << " ";              }
std::cout<<std::endl;

   float *d_data, *d_sum, *d_min, *d_max;
   int numBlocks = (N + THREADS_PER_BLOCK - 1) / THREADS_PER_BLOCK;
std::cout<<"Number of Blocks : "<<numBlocks;

cudaMalloc((void**)&d_data, N * sizeof(float));
cudaMalloc((void**)&d_sum, numBlocks * sizeof(float));
cudaMalloc((void**)&d_min, numBlocks * sizeof(float));
cudaMalloc((void**)&d_max, numBlocks * sizeof(float));

cudaMemcpy(d_data, h_data, N * sizeof(float), cudaMemcpyHostToDevice);

computeSumMinMax<<<numBlocks, THREADS_PER_BLOCK>>>(d_data, d_sum, d_min,
d_max);
cudaDeviceSynchronize();

cudaError_t error = cudaGetLastError();
   if (error != cudaSuccess) {
std::cout<< "CUDA Error: " <<cudaGetErrorString(error) <<std::endl;
   }

   float *h_sum = new float[numBlocks];
   float *h_min = new float[numBlocks];
   float *h_max = new float[numBlocks];

cudaMemcpy(h_sum, d_sum, numBlocks * sizeof(float), cudaMemcpyDeviceToHost);
cudaMemcpy(h_min, d_min, numBlocks * sizeof(float), cudaMemcpyDeviceToHost);
cudaMemcpy(h_max, d_max, numBlocks * sizeof(float), cudaMemcpyDeviceToHost);
```

```
std::cout<< "Block results : " <<std::endl;
   for (int i = 0; i<numBlocks; ++i) {
std::cout<< "Block " <<i<< " -> Sum: " <<h_sum[i] << ", Min: " <<h_min[i] << ", Max: "
<<h_max[i] <<std::endl;
   }

   float totalSum = 0.0f;        float finalMin = h_min[0];     float finalMax = h_max[0];

   for (int i = 0; i<numBlocks; ++i) {
totalSum += h_sum[i];
finalMin = std::min(finalMin, h_min[i]);
finalMax = std::max(finalMax, h_max[i]);
   }

std::cout<< "Final Sum: " <<totalSum<<std::endl;
std::cout<< "Final Min: " <<finalMin<<std::endl;
std::cout<< "Final Max: " <<finalMax<<std::endl;

delete[] h_data;        delete[] h_sum;        delete[] h_min;        delete[] h_max;
cudaFree(d_data);       cudaFree(d_sum);       cudaFree(d_min);       cudaFree(d_max);

   return 0;              }
EOF

#RUN
./vector_minmaxsum
```

Output:

Initial data sample: 383 886 777 915 793 335 386 492 649 421 362 27 690 59 763 926 540 426 172 736 211 368 567 429 782
Number of Blocks : 4
Block results :
Block 0 -> Sum: 4967, Min: 335, Max: 915
Block 1 -> Sum: 3897, Min: 27, Max: 926
Block 2 -> Sum: 3449, Min: 172, Max: 736
Block 3 -> Sum: 782, Min: 782, Max: 782
Final Sum: 13095 Final Min: 27 Final Max: 926

मौलाना आज़ाद राष्ट्रीय प्रौद्योगिकी संस्थान, भोपाल – 462003
MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY, BHOPAL – 462003

कंप्यूटर विज्ञान एवं अभियांत्रिकी विभाग
(Department of Computer Science and Engineering)

LAB: 05

1. मैट्रिक्स मल्टिप्लिकेशन के लिए कूड़ा सी में थ्रेड्स का उपयोग करके यूजर ड्रिवन प्रोग्राम इम्प्लीमेंट करें। इनपुट के रूप में मैट्रिक्स का आकार (rows एवं columns की संख्या) लें। सीपीयू पर समान कार्य के लिए आवश्यक समय की तुलना करें।

Implement a user-driven program i n CUDA C for matrix multiplication using the concept of threads. Take size of matrix (number of rows and columns) as input. Compute the time required for same task on CPU.

```
%%shell
nvcc -arch=sm_70 -o mat_mul -x cu - <<'EOF'

#include <iostream>
#include <cuda_runtime.h>
#define N 8

__global__ void matrixMulKernel(float *A, float *B, float *C, int width) {
   int row = threadIdx.y + blockDim.y * blockIdx.y;
   int col = threadIdx.x + blockDim.x * blockIdx.x;

   if (row < width && col < width) {
      float sum = 0.0f;
      for (int k = 0; k < width; ++k) {
         sum += A[row * width + k] * B[k * width + col];              }
C[row * width + col] = sum;            }                  }

void fillMatrix1(float *mat, int size) {
   for (int i = 0; i< size; i++) {                  mat[i] = 2;              }              }

void fillMatrix2(float *mat, int size) {
   for (int i = 0; i< size; i++) {                  mat[i] = 3;              }              }

void printMatrix(float *mat, int width) {
   for (int i = 0; i< width; ++i) {
      for (int j = 0; j < width; ++j) {    std::cout<<mat[i * width + j] << "\t";}
```

```
std::cout<< "\n";                }                }
int main() {
   int size = N * N;             int bytes = size * sizeof(float);

   float *h_A = new float[size];       float *h_B = new float[size];  float *h_C = new float[size];

   fillMatrix1(h_A, size);             fillMatrix2(h_B, size);

   float *d_A, *d_B, *d_C;
cudaMalloc(&d_A, bytes);    cudaMalloc(&d_B, bytes);    cudaMalloc(&d_C, bytes);

cudaMemcpy(d_A, h_A, bytes, cudaMemcpyHostToDevice);
cudaMemcpy(d_B, h_B, bytes, cudaMemcpyHostToDevice);

   dim3 threadsPerBlock(4, 4);
   dim3 blocksPerGrid((N + 3) / 4, (N + 3) / 4);

matrixMulKernel<<<blocksPerGrid, threadsPerBlock>>>(d_A, d_B, d_C, N);
cudaDeviceSynchronize();

cudaError_t error = cudaGetLastError();
   if (error != cudaSuccess) {
std::cout<< "CUDA Error: " <<cudaGetErrorString(error) <<std::endl;
   }

cudaMemcpy(h_C, d_C, bytes, cudaMemcpyDeviceToHost);

std::cout<< "Matrix A:\n";                printMatrix(h_A, N);
std::cout<< "\nMatrix B:\n";               printMatrix(h_B, N);
std::cout<< "\nMatrix C (Result of A x B):\n";      printMatrix(h_C, N);

delete[] h_A;          delete[] h_B;          delete[] h_C;

cudaFree(d_A);              cudaFree(d_B);              cudaFree(d_C);

   return 0;           }
EOF
```

RUN
./mat_mul

Output:

Matrix A:

2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2

Matrix B:

3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3
3	3	3	3	3	3	3	3

Matrix C (Result of A x B):

48	48	48	48	48	48	48	48
48	48	48	48	48	48	48	48
48	48	48	48	48	48	48	48
48	48	48	48	48	48	48	48
48	48	48	48	48	48	48	48
48	48	48	48	48	48	48	48
48	48	48	48	48	48	48	48
48	48	48	48	48	48	48	48

---

2. मैट्रिक्स मल्टिप्लिकेशन के लिए कूड़ा सी में थ्रेड्स एवं शेयर्ड मेमोरी का उपयोग करके यूजर ड्रिवन प्रोग्राम इम्प्लीमेंट करें। इनपुट के रूप में मैट्रिक्स का आकार (rows एवं columns की संख्या) लें। सीपीयू पर समान कार्य के लिए आवश्यक समय की तुलना करें।

Implement a user-driven program i n CUDA C for matrix multiplication using the concept of threads and shared memory. Take size of matrix (number of rows and columns) as input. Compute the time required for same task on CPU.

```
%%shell
nvcc -arch=sm_70 -o tile_mat_mul -x cu - <<'EOF'

#include <iostream>
#include <cstdlib>
#include <cuda_runtime.h>

#define N 8
#define TILE_SIZE 4

__global__ void matrixMulShared(float *A, float *B, float *C, int width) {
  __shared__ float tileA[TILE_SIZE][TILE_SIZE];
  __shared__ float tileB[TILE_SIZE][TILE_SIZE];

  int row = threadIdx.y + blockDim.y * blockIdx.y;
  int col = threadIdx.x + blockDim.x * blockIdx.x;
  float value = 0;

  for (int t = 0; t < (width + TILE_SIZE - 1) / TILE_SIZE; ++t) {
    if (row < width && t * TILE_SIZE + threadIdx.x< width)
tileA[threadIdx.y][threadIdx.x] = A[row * width + t * TILE_SIZE + threadIdx.x];
    else
tileA[threadIdx.y][threadIdx.x] = 0.0;

    if (col < width && t * TILE_SIZE + threadIdx.y< width)
tileB[threadIdx.y][threadIdx.x] = B[(t * TILE_SIZE + threadIdx.y) * width + col];
    else
tileB[threadIdx.y][threadIdx.x] = 0.0;
```

```
    __syncthreads();

    for (int i = 0; i< TILE_SIZE; ++i) {
      value += tileA[threadIdx.y][i] * tileB[i][threadIdx.x];
    }

    __syncthreads();
  }

  if (row < width && col < width)
C[row * width + col] = value;
}

void fillMatrix1(float *matrix, int size) {
  for (int i = 0; i< size; ++i)          matrix[i] = 4;          }

void fillMatrix2(float *matrix, int size) {
  for (int i = 0; i< size; ++i)          matrix[i] = 5;          }

void printMatrix(float *matrix, int width) {
  for (int i = 0; i< width; ++i) {
    for (int j = 0; j < width; ++j) {
std::cout<<matrix[i * width + j] << "\t";              }
std::cout<< "\n";              }              }

int main() {
  int size = N * N;          int bytes = size * sizeof(float);

  float *h_A = new float[size];              float *h_B = new float[size];
  float *h_C = new float[size];

  fillMatrix1(h_A, size);              fillMatrix2(h_B, size);

  float *d_A, *d_B, *d_C;
cudaMalloc(&d_A, bytes);              cudaMalloc(&d_B, bytes);
cudaMalloc(&d_C, bytes);
```

```
cudaMemcpy(d_A, h_A, bytes, cudaMemcpyHostToDevice);
cudaMemcpy(d_B, h_B, bytes, cudaMemcpyHostToDevice);

   dim3 threads(TILE_SIZE, TILE_SIZE);
   dim3 blocks((N + TILE_SIZE - 1) / TILE_SIZE, (N + TILE_SIZE - 1) / TILE_SIZE);

matrixMulShared<<<blocks, threads>>>(d_A, d_B, d_C, N);
cudaDeviceSynchronize();

cudaError_t error = cudaGetLastError();
   if (error != cudaSuccess) {
std::cout<< "CUDA Error: " <<cudaGetErrorString(error) <<std::endl;
   }

cudaMemcpy(h_C, d_C, bytes, cudaMemcpyDeviceToHost);

std::cout<< "Matrix A:\n";                    printMatrix(h_A, N);
std::cout<< "\nMatrix B:\n";                  printMatrix(h_B, N);
std::cout<< "\nMatrix C = A x B:\n";                printMatrix(h_C, N);

delete[] h_A;          delete[] h_B;                delete[] h_C;
cudaFree(d_A);                cudaFree(d_B);                cudaFree(d_C);

   return 0;              }              EOF

#RUN
./tile_mat_mul
```

Output:

```
Matrix A:
4       4       4       4       4       4       4       4
4       4       4       4       4       4       4       4
4       4       4       4       4       4       4       4
4       4       4       4       4       4       4       4
4       4       4       4       4       4       4       4
4       4       4       4       4       4       4       4
4       4       4       4       4       4       4       4
4       4       4       4       4       4       4       4
```

Matrix B:

5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5
5	5	5	5	5	5	5	5

Matrix C = A x B:

160	160	160	160	160	160	160	160
160	160	160	160	160	160	160	160
160	160	160	160	160	160	160	160
160	160	160	160	160	160	160	160
160	160	160	160	160	160	160	160
160	160	160	160	160	160	160	160
160	160	160	160	160	160	160	160
160	160	160	160	160	160	160	160

मौलाना आज़ाद राष्ट्रीय प्रौद्योगिकी संस्थान, भोपाल – 462003
MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY, BHOPAL – 462003

कंप्यूटर विज्ञान एवं अभियांत्रिकी विभाग
(Department of Computer Science and Engineering)

LAB: 06

1. वर्ड काउंट प्रॉब्लम के लिए कूड़ा सी में प्रोग्राम इम्प्लीमेंट करें। प्रोग्राम एक फाइल रीड करेगा और फाइल में मौजूद यूनिक वर्ड्स एवं उनके काउंट को ज्ञात करेगा।

Implement a program in CUDA C for Word Count Problem. The program must READ a File, determine the distinct words and count the occurrence of each word present in the file.

```
%%shell
nvcc -arch=sm_70 -o word_count -x cu - <<'EOF'

#include <stdio.h>
#include <string.h>

#define MAX_WORDS 100
#define MAX_WORD_LEN 20

__device__ int compare_words(const char* a, const char* b) {
   for (int i = 0; i< MAX_WORD_LEN; i++) {
      if (a[i] != b[i]) return 0;
      if (a[i] == '\0') break;                }
   return 1;                }

__global__ void count_words_kernel(char words[MAX_WORDS][MAX_WORD_LEN], int *word_count, int total_words) {
   int i = blockIdx.x * blockDim.x + threadIdx.x;

   if (i<total_words) {
      int found = 0;
      for (int j = 0; j <i; j++) {
         if (compare_words(words[i], words[j])) {
            found = 1;              break;            }                }

      if (!found) {
         int count = 1;
         for (int k = i + 1; k <total_words; k++) {
```

```
            if (compare_words(words[i], words[k])) {
               count++;          }                }
word_count[i] = count;                  }
         else {            word_count[i] = 0;              }
   }              }

int main() {
   const char *input = "cuda is fun and cuda is fast and parallel";
   char host_words[MAX_WORDS][MAX_WORD_LEN];
   int word_count[MAX_WORDS] = {0};

   int total_words = 0;
   char buffer[strlen(input) + 1];
strcpy(buffer, input);

   char *token = strtok(buffer, " ");
   while (token != NULL &&total_words< MAX_WORDS) {
strncpy(host_words[total_words], token, MAX_WORD_LEN);
host_words[total_words][MAX_WORD_LEN - 1] = '\0';
total_words++;
      token = strtok(NULL, " ");                }

   char (*d_words)[MAX_WORD_LEN];
   int *d_counts;

cudaMalloc((void**)&d_words, sizeof(char) * MAX_WORDS * MAX_WORD_LEN);
cudaMalloc((void**)&d_counts, sizeof(int) * MAX_WORDS);

cudaMemcpy(d_words, host_words, sizeof(char) * MAX_WORDS * MAX_WORD_LEN,
cudaMemcpyHostToDevice);

   int threadsPerBlock = 4;
   int blocks = (total_words + threadsPerBlock - 1) / threadsPerBlock;

count_words_kernel<<<blocks, threadsPerBlock>>>(d_words, d_counts, total_words);
cudaDeviceSynchronize();
```

```
cudaMemcpy(word_count, d_counts, sizeof(int) * MAX_WORDS,
cudaMemcpyDeviceToHost);

printf("Distinct word counts:\n");
   for (int i = 0; i<total_words; i++) {
      if (word_count[i] > 0) {
printf("%s: %d\n", host_words[i], word_count[i]);
      }          }

cudaFree(d_words);          cudaFree(d_counts);

   return 0;          }
EOF

#RUN
./word_count
```

Output:

```
Distinct word counts:
cuda: 2
is: 2
fun: 1
and: 2
fast: 1
parallel: 1
```

मौलाना आज़ाद राष्ट्रीय प्रौद्योगिकी संस्थान, भोपाल – 462003
MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY, BHOPAL – 462003

कंप्यूटर विज्ञान एवं अभियांत्रिकी विभाग
(Department of Computer Science and Engineering)

LAB: 07

1. मर्ज सॉर्ट अल्गोरिथम का उपयोग करके कूड़ा सी में पैरेलल सॉर्टिंग का प्रोग्राम इम्प्लीमेंट करें।

Implement a program in CUDA C for Parallel Sorting using Merge Sort algorithm.

```
%%shell
nvcc -arch=sm_70 -o merge_sort_parallel -x cu - <<'EOF'

#include <stdio.h>
#include <stdlib.h>
#include <cuda.h>

#define N 25
#define THREADS_PER_BLOCK 4

__device__ void merge(int *input, int *output, int left, int mid, int right) {
   int i = left;            int j = mid + 1;                  int k = left;

   while (i<= mid && j <= right) {
      if (input[i] <= input[j])          output[k++] = input[i++];
      else                               output[k++] = input[j++];          }

   while (i<= mid) output[k++] = input[i++];
   while (j <= right) output[k++] = input[j++];                }

__global__ void merge_pass(int *input, int *output, int width, int size) {
   int idx = blockIdx.x * blockDim.x + threadIdx.x;
   int left = idx * (2 * width);          int mid = left + width - 1;
   int right = left + 2 * width - 1;

   if (right >= size) {          right = size - 1;                }

   if (left < size && mid < size)
merge(input, output, left, mid, right);                }
```

```
void merge_sort_gpu(int *data) {
   int *d_input, *d_output;                    size_t size = N * sizeof(int);
cudaMalloc((void**)&d_input, size);cudaMalloc((void**)&d_output, size);
cudaMemcpy(d_input, data, size, cudaMemcpyHostToDevice);

   int *temp;
   for (int width = 1; width < N; width *= 2) {
      int numBlocks = (N / (2 * width)) + 1;
merge_pass<<<numBlocks, THREADS_PER_BLOCK>>>(d_input, d_output, width, N);
cudaDeviceSynchronize();

      temp = d_input;  d_input = d_output;          d_output = temp;       }

cudaMemcpy(data, d_input, size, cudaMemcpyDeviceToHost);

cudaFree(d_input);          cudaFree(d_output);          }

int main() {
   int arr[N];

printf("Unsorted array:\n");
   for (int i = 0; i< N; i++) {
arr[i] = rand() % 100;          printf("%d ", arr[i]);          }
printf("\n");

merge_sort_gpu(arr);

printf("Sorted array:\n");
   for (int i = 0; i< N; i++) {
printf("%d ", arr[i]);          }
printf("\n");

   return 0;          }
EOF

#RUN
./merge_sort_parallel
```

Output:

Unsorted array:

83	86	77	15	93	35	86	92	49	21	62	27	90
59	63	26	40	26	72	36	11	68	67	29	82	

Sorted array:

11	15	21	26	26	27	29	35	36	40	49	59	62
63	67	68	72	77	82	83	86	86	90	92	93	

---

2. कूड़ा सी में पैरेलल सर्चिंग के लिए प्रोग्राम इम्प्लीमेंट करें।

Implement a program in CUDA C for Parallel Searching.

```
%%shell
nvcc -arch=sm_70 -o search_parallel -x cu - <<'EOF'

#include <stdio.h>
#include <stdlib.h>
#include <cuda.h>

#define N 16

__global__ void parallel_search(int *arr, int target, int *result) {
   int idx = blockIdx.x * blockDim.x + threadIdx.x;
   if (idx< N) {
      if (arr[idx] == target) {
atomicMin(result, idx);  // Find the smallest index if multiple matches
      }          }                }

int main() {
   int h_arr[N], h_result;                int target = 25;

   for (int i = 0; i< N; i++) {           h_arr[i] = rand() % 100;              }

h_arr[4] = target;
printf("Elements of List : \n");
for(int i = 0; i< N; i++) {               printf("%d ", h_arr[i]);              }
printf("\n");
```

```
   int *d_arr, *d_result;                    size_t size = N * sizeof(int);

cudaMalloc((void **)&d_arr, size);   cudaMalloc((void **)&d_result, sizeof(int));

cudaMemcpy(d_arr, h_arr, size, cudaMemcpyHostToDevice);

   int init_val = N;
cudaMemcpy(d_result, &init_val, sizeof(int), cudaMemcpyHostToDevice);

   int threadsPerBlock = 4;
   int blocks = (N + threadsPerBlock - 1) / threadsPerBlock;
parallel_search<<<blocks, threadsPerBlock>>>(d_arr, target, d_result);
cudaDeviceSynchronize();

cudaMemcpy(&h_result, d_result, sizeof(int), cudaMemcpyDeviceToHost);

   if (h_result< N) {
printf("Target %d found at index: %d\n", target, h_result);          }
   else {
printf("Target %d not found in the array.\n", target);               }

cudaFree(d_arr);              cudaFree(d_result);

   return 0;             }
EOF

#RUN
./search_parallel
```

Output:

```
Elements of List :
83      86      77      15      25      35      86      92      49      21
62      27      90      59      63      26
Target 25 found at index: 4
```

मौलाना आज़ाद राष्ट्रीय प्रौद्योगिकी संस्थान, भोपाल – 462003
MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY, BHOPAL – 462003

कंप्यूटर विज्ञान एवं अभियांत्रिकी विभाग
(Department of Computer Science and Engineering)

LAB: 08

1. मैट्रिक्स मल्टिप्लिकेशन के लिए मैप रिड्यूस प्रोग्राम इम्प्लीमेंट करे।

Implement a MapReduce program to implement Matrix Multiplication.

```
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;

public class MatrixMultiply {
   public static class MatrixMapper extends Mapper<Object, Text, Text, Text> {
      int ncolsB;                  int nrowsA;

      @Override
      protected void setup(Context context) throws IOException, InterruptedException {
         Configuration conf = context.getConfiguration();
ncolsB = conf.getInt("ncolsB", 0);              nrowsA = conf.getInt("nrowsA", 0);
      }

      @Override
      public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
String[] parts = value.toString().split(",");
         if (parts.length != 4) return;
```

```
        String matrixName = parts[0].trim();
        int row = Integer.parseInt(parts[1].trim());
        int col = Integer.parseInt(parts[2].trim());
        double val = Double.parseDouble(parts[3].trim());

        if (matrixName.equalsIgnoreCase("A")) {
           for (int j = 0; j <ncolsB; j++) {
              Text outKey = new Text(row + "," + j);
              Text outVal = new Text("A," + col + "," + val);
context.write(outKey, outVal);                    }
        } else if (matrixName.equalsIgnoreCase("B")) {
           for (int i = 0; i<nrowsA; i++) {
              Text outKey = new Text(i + "," + col);
        Text outVal = new Text("B," + row + "," + val);
context.write(outKey, outVal);}
        }                  }                  }

   public static class MatrixReducer extends Reducer<Text, Text, Text, Text> {
      @Override
      public void reduce(Text key, Iterable<Text> values, Context context)
            throws IOException, InterruptedException {
         HashMap<Integer, List<Double>>mapA = new HashMap<>();
         HashMap<Integer, List<Double>>mapB = new HashMap<>();

         for (Text val : values) {
String[] parts = val.toString().split(",");
            String tag = parts[0].trim();
            int k = Integer.parseInt(parts[1].trim());
            double v = Double.parseDouble(parts[2].trim());

            if (tag.equals("A")) {
               if (!mapA.containsKey(k))                  mapA.put(k, new ArrayList<Double>());
mapA.get(k).add(v);
            } else if (tag.equals("B")) {
               if (!mapB.containsKey(k))                  mapB.put(k, new ArrayList<Double>());
mapB.get(k).add(v);            }                  }
         double sum = 0.0;
```

```
        for (Integer k :mapA.keySet()) {
          if (mapB.containsKey(k)) {
            for (double aVal : mapA.get(k)) {
              for (double bVal : mapB.get(k)) {
                sum += aVal * bVal;        }              }              }              }
context.write(key, new Text(Double.toString(sum)));              }              }

  public static void main(String[] args) throws Exception {
    if (args.length != 3) {
System.err.println("Usage: MatrixMultiply<input path><output path><config>");
System.err.println("Example config: nrowsA=2,ncolsB=2");
System.exit(-1);              }
    Configuration conf = new Configuration();
String[] configParts = args[2].split(",");
    for (String part :configParts) {
String[] keyVal = part.split("=");
conf.setInt(keyVal[0].trim(), Integer.parseInt(keyVal[1].trim()));              }

    Job job = Job.getInstance(conf, "Matrix Multiply");
job.setJarByClass(MatrixMultiply.class);
job.setMapperClass(MatrixMapper.class);
job.setReducerClass(MatrixReducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(Text.class);

FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));

System.exit(job.waitForCompletion(true) ?0 : 1);              }              }
```

Output:

matrix_input.txt:

```
A,0,0,1        A,0,1,2        A,1,0,3        A,1,1,4
B,0,0,5        B,0,1,6        B,1,0,7        B,1,1,8
```

Final HDFS output:

```
0,0    19.0            0,1    22.0            1,0    43.0            1,1    50.0
```

मौलाना आज़ाद राष्ट्रीय प्रौद्योगिकी संस्थान, भोपाल – 462003
MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY, BHOPAL – 462003

कंप्यूटर विज्ञान एवं अभियांत्रिकी विभाग
(Department of Computer Science and Engineering)

LAB: 09

1. टेक्स्ट डेटासेट में यूनिक शब्दों की गणना करने के लिए मैप रिड्यूस प्रोग्राम इम्प्लीमेंट करें।

Implement a MapReduce program to calculate the word count in a text dataset.

```
import java.io.IOException;
import java.util.StringTokenizer;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class WordCount {
   public static class TokenizerMapperextends Mapper<Object, Text, Text, IntWritable> {
      private final static IntWritable one = new IntWritable(1);
      private Text word = new Text();

      public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
StringTokenizeritr = new StringTokenizer(value.toString());
         while (itr.hasMoreTokens()) {
word.set(itr.nextToken().toLowerCase());
context.write(word, one);              }              }              }

   public static class IntSumReducer
      extends Reducer<Text, IntWritable, Text, IntWritable> {
```

```
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values,Context context) throws
IOException, InterruptedException {
        int sum = 0;
        for (IntWritableval : values) {                sum += val.get();                }
result.set(sum);
context.write(key, result);            }                }

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");

job.setJarByClass(WordCount.class);                job.setMapperClass(TokenizerMapper.class);
job.setCombinerClass(IntSumReducer.class);         job.setReducerClass(IntSumReducer.class);
job.setOutputKeyClass(Text.class);            job.setOutputValueClass(IntWritable.class);

FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));

System.exit(job.waitForCompletion(true) ?0 : 1);            }            }
```

Output:

Input File: Word Count.txt

This is a test This is another test MapReduce is a powerful tool
Hadoop and MapReduce work together This is a simple example
Learning MapReduce is fun
Word count is a common MapReduce example
MapReduce can handle big data Data processing is essential
MapReduce makes distributed processing easier

Final HDFS output:

a 4	and 1	another 1	big 1
can 1	common 1	count 1	data 2
distributed 1	easier 1	example 2	fun 1
handle 1	hadoop 1	is 6	learning 1
makes 1	mapreduce 5	processing 2	powerful 1
simple 1	test 2	this 3	together 1
tool1	word 1	work 1	

मौलाना आज़ाद राष्ट्रीय प्रौद्योगिकी संस्थान, भोपाल – 462003
MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY, BHOPAL – 462003

कंप्यूटर विज्ञान एवं अभियांत्रिकी विभाग
(Department of Computer Science and Engineering)

LAB: 10

1. उपस्थिति लॉग से प्रत्येक कमिचारी द्वारा काम किये गए कुल घंटों की गणना करने के लिए मैप रिड्यूस प्रोग्राम इम्प्लीमेंट करें।

Implement a MapReduce program to calculate the total hours worked by each employee from attendance logs.

```
import java.io.IOException;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class EmployeeHours {
   public static class TokenizerMapperextends Mapper<Object, Text, Text, IntWritable> {
      private final static IntWritable hours = new IntWritable();
      private Text employee = new Text();

      public void map(Object key, Text value, Context context) throws IOException, InterruptedException
{
String[] parts = value.toString().split("\t");
         if (parts.length == 3) {
employee.set(parts[0].trim()); // Employee name
hours.set(Integer.parseInt(parts[2].trim())); // Hours Worked
context.write(employee, hours);          }                  }                  }

   public static class SumReducerextends Reducer<Text, IntWritable, Text, IntWritable> {
      private IntWritable result = new IntWritable();
      public void reduce(Text key, Iterable<IntWritable> values,Context context) throws IOException,
InterruptedException {
         int sum = 0;
         for (IntWritableval : values) {              sum += val.get();                  }
```

```
result.set(sum);            context.write(key, result);                }              }
   public static void main(String[] args) throws Exception {
      Configuration conf = new Configuration();
      Job job = Job.getInstance(conf, "employee total hours");
job.setJarByClass(EmployeeHours.class);
job.setMapperClass(TokenizerMapper.class);
job.setCombinerClass(SumReducer.class);
job.setReducerClass(SumReducer.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(IntWritable.class);
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
System.exit(job.waitForCompletion(true) ?0 : 1);
   }            }
```

Output:

Input file: attendence_logs.xlsx

Final HDSF output:

```
Alice   63
Bob     60
John    59
Mary    60
Steve   82
```

मौलाना आज़ाद राष्ट्रीय प्रौद्योगिकी संस्थान, भोपाल – 462003
MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY, BHOPAL – 462003

कंप्यूटर विज्ञान एवं अभियांत्रिकी विभाग
(Department of Computer Science and Engineering)

LAB: 11

1. प्रत्येक विषय में अंकों के आधार पर शीर्ष 3 छात्रों को खोजने के लिए मैप रिड्यूस प्रोग्राम इम्प्लीमेंट करें।

Implement a MapReduce program to find the top 3 students in each subject based on their scores.

```
import java.io.IOException;
import java.util.PriorityQueue;
import java.util.Comparator;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class Top3StudentsPerSubject {
   public static class ScoreMapper extends Mapper<Object, Text, Text, Text> {
      private Text subject = new Text();
      private Text studentScore = new Text();

      public void map(Object key, Text value, Context context) throws IOException, InterruptedException
{
String[] parts = value.toString().split("\t");
         if (parts.length == 3 && !parts[0].equals("Student")) {
            String student = parts[0].trim();
            String subj = parts[1].trim();              String score = parts[2].trim();
subject.set(subj);
studentScore.set(student + "," + score);
context.write(subject, studentScore);               }                   }                   }

   public static class Top3Reducer extends Reducer<Text, Text, Text, Text> {
      private Text result = new Text();
      public void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
InterruptedException {
```

```
PriorityQueue<String[]>topStudents = new PriorityQueue<>(3, Comparator.comparingInt(a ->Integer.parseInt(a[1])));
        for (Text val : values) {
String[] tokens = val.toString().split(",");
            if (tokens.length != 2) continue;
            String student = tokens[0];
            int score = Integer.parseInt(tokens[1]);

topStudents.add(new String[]{student, String.valueOf(score)});
            if (topStudents.size() > 3) {                topStudents.poll();                }                }

PriorityQueue<String[]> sorted = new PriorityQueue<>(3, (a, b) ->Integer.parseInt(b[1]) - Integer.parseInt(a[1]));
sorted.addAll(topStudents);

        StringBuilder sb = new StringBuilder();
        while (!sorted.isEmpty()) {
String[] top = sorted.poll();
sb.append(top[0]).append(" (").append(top[1]).append("), ");                }

        String output = sb.toString().replaceAll(", $", "");
result.set(output);
context.write(key, result);                }                }

   public static void main(String[] args) throws Exception {
      Configuration conf = new Configuration();
      Job job = Job.getInstance(conf, "Top 3 Students Per Subject");

job.setJarByClass(Top3StudentsPerSubject.class);
job.setMapperClass(ScoreMapper.class);
job.setReducerClass(Top3Reducer.class);

job.setMapOutputKeyClass(Text.class);
job.setMapOutputValueClass(Text.class);
job.setOutputKeyClass(Text.class);
job.setOutputValueClass(Text.class);

FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));

System.exit(job.waitForCompletion(true) ?0 : 1);                }                }
```

Output:

Input file: student_scores.xlsx

Final HDFS output:

English Sophia (100), Mary (87), John (86)
Geography Alice (90), Sophia (89), John (79)
History Alice (94), Mary (92), Steve (91)
Maths Alice (93), Olivia (87), Emma (89)
Science Bob (93), Steve (91), Sophia (91)

मौलाना आज़ाद राष्ट्रीय प्रौद्योगिकी संस्थान, भोपाल – 462003
MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY, BHOPAL – 462003

कंप्यूटर विज्ञान एवं अभियांत्रिकी विभाग
(Department of Computer Science and Engineering)

LAB: 12

1. प्रत्येक देश में सबसे अधिक बिकने वाले उत्पाद का निर्धारण करने के लिए मैप रिड्यूस प्रोग्राम इम्प्लीमेंट करें।

Implement a MapReduce program to determine the most frequently sold product based in each country.

```
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class TopProductPerCountry {
   public static class CountryProductMapper extends Mapper<Object, Text, Text, Text> {
      private Text country = new Text();
      private Text product = new Text();

      public void map(Object key, Text value, Context context) throws IOException, InterruptedException
{
String[] cols = value.toString().split("\t");
         if (cols.length>= 2 && !cols[0].equalsIgnoreCase("Product")) {
product.set(cols[0].trim());                    country.set(cols[1].trim());
context.write(country, product); }                    }                    }

   public static class MostFrequentReducer extends Reducer<Text, Text, Text, Text> {
      private Text result = new Text();
      public void reduce(Text key, Iterable<Text> values, Context context) throws IOException,
InterruptedException {
         Map<String, Integer>productCount = new HashMap<>();

         for (Text val : values) {
            String product = val.toString();
```

```
productCount.put(product, productCount.getOrDefault(product, 0) + 1);
        }
        String topProduct = null;
        int maxCount = 0;
        for (Map.Entry<String, Integer>entry :productCount.entrySet()) {
          if (entry.getValue() >maxCount) {
topProduct = entry.getKey();      maxCount = entry.getValue();              }         }

result.set(topProduct + " (" + maxCount + " sales)");
context.write(key, result);                    }                    }

   public static void main(String[] args) throws Exception {
     Configuration conf = new Configuration();
     Job job = Job.getInstance(conf, "Top Product Per Country");

job.setJarByClass(TopProductPerCountry.class);
job.setMapperClass(CountryProductMapper.class);
job.setReducerClass(MostFrequentReducer.class);

job.setMapOutputKeyClass(Text.class);          job.setMapOutputValueClass(Text.class);
job.setOutputKeyClass(Text.class);             job.setOutputValueClass(Text.class);

FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));

System.exit(job.waitForCompletion(true) ?0 : 1);                    }                    }
```

Output:

Input file: SalesJan2009.xlsx

Final HDSF output:

United States	Product1 (8 sales)	United Kingdom	Product1 (3 sales)
Ireland	Product1 (2 sales)	Australia	Product1 (1 sales)
India	Product1 (1 sales)	Canada	Product1 (1 sales)
France	Product1 (1 sales)	Netherlands	Product1 (1 sales)
South Africa	Product1 (1 sales)	Finland	Product1 (1 sales)

मौलाना आज़ाद राष्ट्रीय प्रौद्योगिकी संस्थान, भोपाल – 462003
MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY, BHOPAL – 462003

कंप्यूटर विज्ञान एवं अभियांत्रिकी विभाग
(Department of Computer Science and Engineering)

विषय समन्वयक / Subject Coordinators

डॉ.मितुल कुमार अहीरवाल (Dr. Mitul Kumar Ahirwal)

डॉ.यदुनाथ पाठक (Dr. Yadunath Pathak)

डॉ. मनीषा सिंह (Dr. Manisha Singh)