OS LAB#12

M.Mubin Farid 21K-4827

Task 1)

#include <iostream>

#include <cstdlib>

#include <ctime>

#include <omp.h>

using namespace std;

int main() {

int A[100][100], B[100][100], sum[100][100];

srand(time(NULL));

#pragma omp parallel for

for (int i = 0; i < 100; i++) {

for (int j = 0; j < 100; j++) {

A[i][j] = rand() % 100 + 1;

B[i][j] = rand() % 100 + 1;

}

}

#pragma omp parallel for

for (int i = 0; i < 100; i++) {

for (int j = 0; j < 100; j++) {

sum[i][j] = A[i][j] + B[i][j];

}

}

cout << "Matrix A:" << endl;

for (int i = 0; i < 100; i++) {

for (int j = 0; j < 100; j++) {

cout << A[i][j] << " ";

}

cout << endl;

}

cout << endl;

cout << "Matrix B:" << endl;

for (int i = 0; i < 100; i++) {

for (int j = 0; j < 100; j++) {

cout << B[i][j] << " ";

}

cout << endl;

}

cout << endl;

cout << "Sum of matrices A and B:" << endl;

for (int i = 0; i < 100; i++) {

for (int j = 0; j < 100; j++) {

cout << sum[i][j] << " ";

}

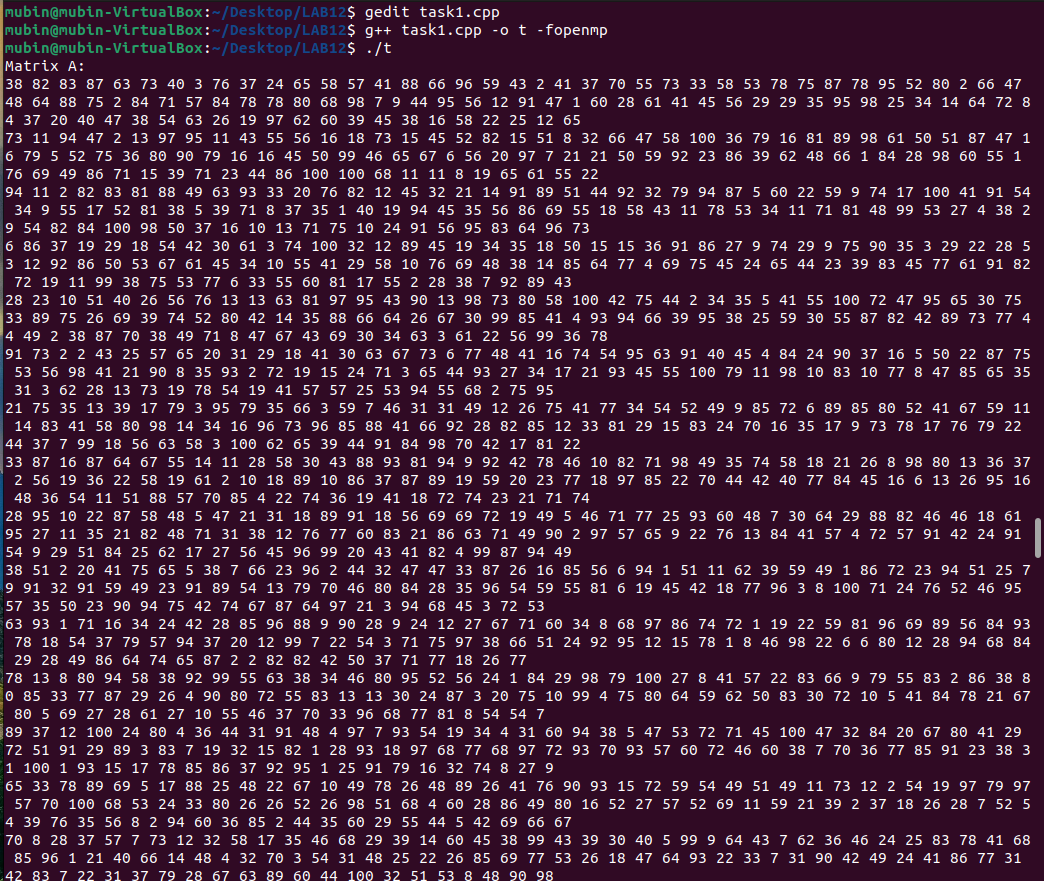
cout << endl;

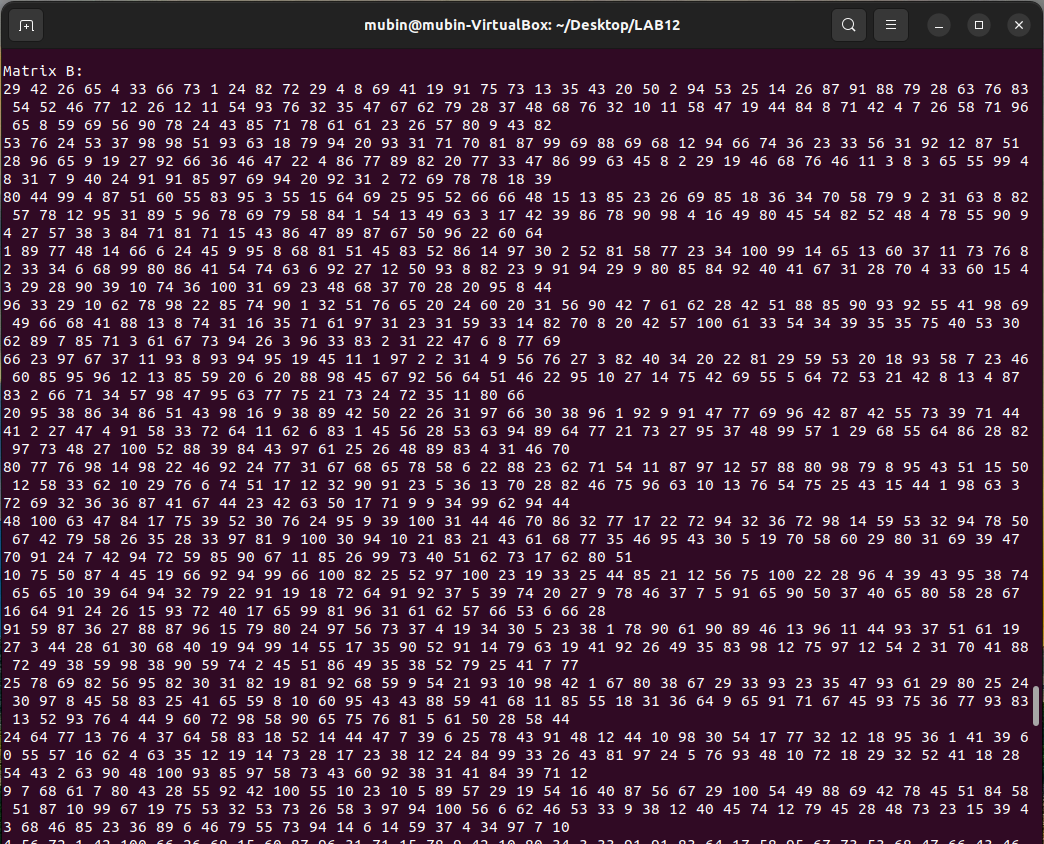
}

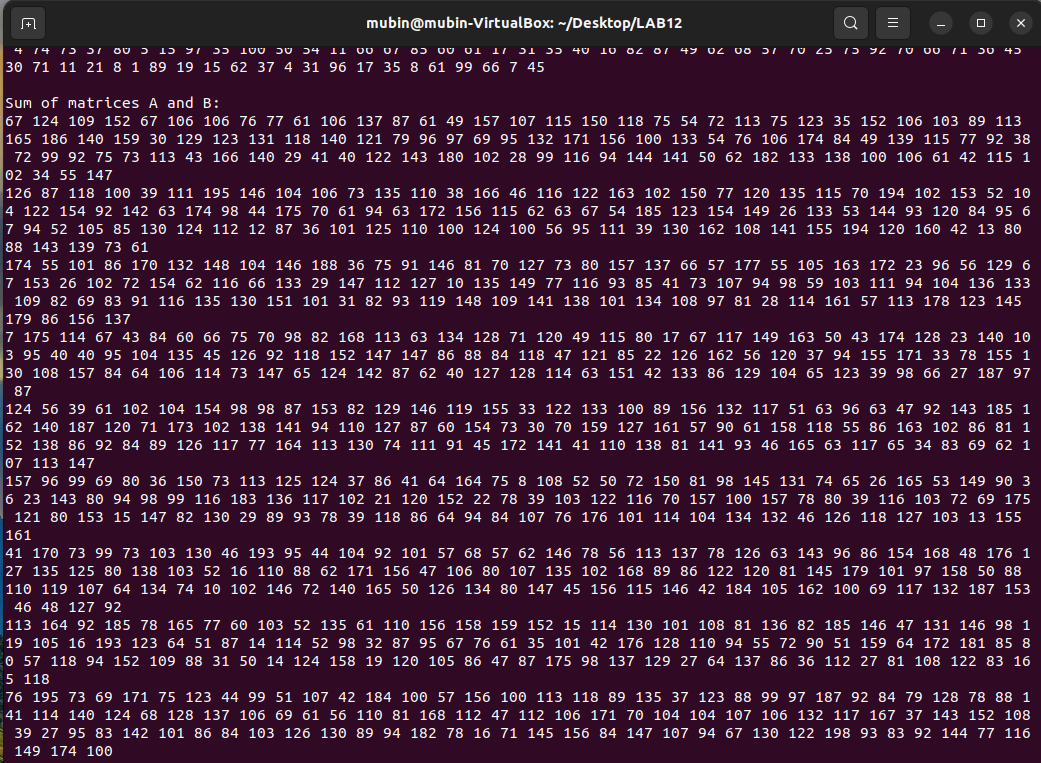
cout << endl;

return 0;

}







Task 2)

#include <iostream>

#include <cstdlib>

#include <ctime>

#include <omp.h>

#define SIZE 10000

using namespace std;

int main() {

int arr[SIZE];

int max\_val = -1;

int max\_index = -1;

int chunksize = SIZE / omp\_get\_max\_threads();

cout << "Making " << omp\_get\_max\_threads() << " chunks of array." << endl;

cout << "Chunk size: " << chunksize << endl;

srand(time(NULL));

for (int i = 0; i < SIZE; i++) {

arr[i] = rand() % 100 + 1;

}

#pragma omp parallel

{

int local\_max\_val = -1;

int local\_max\_index = -1;

#pragma omp for schedule(static, chunksize)

for (int i = 0; i < SIZE; i++) {

if (arr[i] > local\_max\_val) {

local\_max\_val = arr[i];

local\_max\_index = i;

}

}

#pragma omp critical

{

if (local\_max\_val > max\_val) {

max\_val = local\_max\_val;

max\_index = local\_max\_index;

}

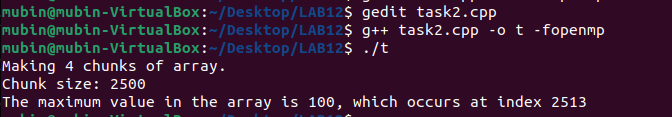
}

}

cout << "The maximum value in the array is " << max\_val << ", which occurs at index " << max\_index << endl;

return 0;

}



Task 3)

#include <iostream>

#include <omp.h>

#include <ctime>

#define ll long long int

using namespace std;

void Bubblesort(int arr[], ll size) {

ll i, j, temp;

bool swapped;

for (i = 0; i < size - 1; i++) {

swapped = false;

#pragma omp parallel for private(j, temp) reduction(||: swapped)

for (j = 0; j < size - i - 1; j++) {

if (arr[j] > arr[j+1]) {

temp = arr[j];

arr[j] = arr[j+1];

arr[j+1] = temp;

swapped = true;

}

}

if (!swapped) break;

}

}

int main() {

ll n=10000;

int arr[n];

srand(time(NULL));

cout<<"Random numbers array:";

for(ll i=0;i<n;i++)

{

arr[i]=rand()%100;

cout<<arr[i]<<" ";

}

Bubblesort(arr,n);

cout << endl;

cout << "Sorted array";

for (int i = 0; i < n; i++) {

cout << arr[i] << " ";

}

cout << endl;

return 0;

}

