

**Class:** Final Year (Computer Science and Engineering)

**Year:** 2023-24

**Semester:** 1

**Course:** High Performance Computing Lab

## Practical No. 5

**Exam Seat No:** 2020BTECS00037

**Title of practical:** Implementation of OpenMP programs.

Implement following Programs using OpenMP with C:

1. Implementation of sum of two lower triangular matrices.
2. Implementation of Matrix-Matrix Multiplication.

### Problem Statement 1:

#### matrix\_sum\_serial

```
#include <stdio.h>
#include <time.h>

// Size of the matrices
#define N 4

int main() {
    // Lower triangular matrices
    int A[N][N];
    int B[N][N];

    // Resultant matrix
    int C[N][N];

    clock_t start_time, end_time;

    // Initialize matrices A and B
    for (int i = 0; i < N; i++) {
        for (int j = 0; j <= i; j++) {
            A[i][j] = i + j + 1;
            B[i][j] = i - j + 1;
        }
    }
}
```

```
start_time = clock();

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

end_time = clock();

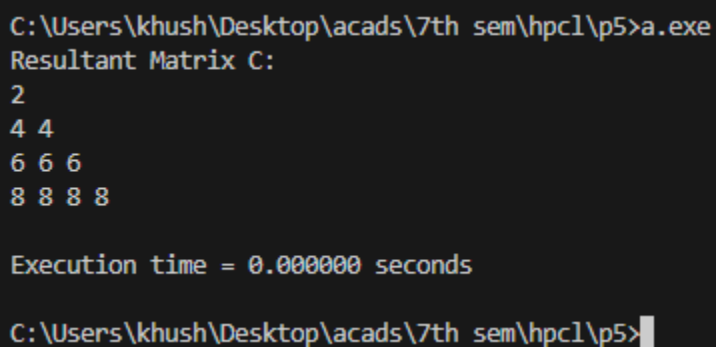
printf("Resultant Matrix C:\n");
for (int i = 0; i < N; i++) {
    for (int j = 0; j <= i; j++) {
        printf("%d ", C[i][j]);
    }
    printf("\n");
}

double exec_time = (double)(end_time - start_time) / CLOCKS_PER_SEC;

printf("\nExecution time = %.6f seconds\n", exec_time);

return 0;
}
```

### Screenshots:



```
C:\Users\khush\Desktop\acads\7th sem\hpc1\p5>a.exe
Resultant Matrix C:
2
4 4
6 6 6
8 8 8 8

Execution time = 0.000000 seconds

C:\Users\khush\Desktop\acads\7th sem\hpc1\p5>
```

## 2. Implementation of Matrix-Matrix Multiplication.

### Problem Statement 2:

#### matrix\_mul\_serial

```
#include <stdio.h>
#include <stdlib.h>
#include <sys/time.h>

#define ROWS_A 8
#define COLS_A 8
#define ROWS_B 8
#define COLS_B 8

double get_seconds(struct timeval start, struct timeval end) {
    return (end.tv_sec - start.tv_sec) + (end.tv_usec - start.tv_usec) / 1e6;
}

int main() {
    srand(time(NULL));

    struct timeval start_time, end_time;

    int A[ROWS_A][COLS_A];
    int B[ROWS_B][COLS_B];
    int C[ROWS_A][COLS_B];

    // Initialize matrices A and B with random values
    for (int i = 0; i < ROWS_A; i++) {
        for (int j = 0; j < COLS_A; j++) {
            // Random values between 0 and 99
            A[i][j] = rand() % 100;
        }
    }

    for (int i = 0; i < ROWS_B; i++) {
        for (int j = 0; j < COLS_B; j++) {
            // Random values between 0 and 99
            B[i][j] = rand() % 100;
        }
    }

    gettimeofday(&start_time, NULL);
```

```
// Perform matrix multiplication
for (int i = 0; i < ROWS_A; i++) {
    for (int j = 0; j < COLS_B; j++) {
        C[i][j] = 0;
        for (int k = 0; k < COLS_A; k++) {
            C[i][j] += A[i][k] * B[k][j];
        }
    }
}

gettimeofday(&end_time, NULL);

// Print the result matrix C
printf("Matrix A:\n");
for (int i = 0; i < ROWS_A; i++) {
    for (int j = 0; j < COLS_A; j++) {
        printf("%d ", A[i][j]);
    }
    printf("\n");
}

printf("\nMatrix B:\n");
for (int i = 0; i < ROWS_B; i++) {
    for (int j = 0; j < COLS_B; j++) {
        printf("%d ", B[i][j]);
    }
    printf("\n");
}

printf("\nResultant Matrix C:\n");
for (int i = 0; i < ROWS_A; i++) {
    for (int j = 0; j < COLS_B; j++) {
        printf("%d ", C[i][j]);
    }
    printf("\n");
}

double exec_time = get_seconds(start_time, end_time);

printf("\nExecution time = %.6f seconds\n", exec_time);

return 0;
}
```

### Screenshots:

```
C:\Users\khusn\Desktop\acadu5\7th Sem\hpc1\p57a.exe
Matrix A:
28 73 79 22 28 55 56 53
33 52 45 71 25 6 35 1
12 3 44 95 11 77 88 92
75 16 95 79 35 43 5 12
50 30 34 23 1 2 95 21
76 49 60 28 10 40 72 22
46 27 90 77 55 80 1 69
71 74 98 37 24 98 72 99

Matrix B:
51 31 60 71 90 11 39 54
15 38 53 34 33 70 86 80
27 67 86 77 94 16 57 39
16 82 98 64 54 25 85 95
78 88 9 1 42 92 38 94
23 96 46 13 45 40 43 33
6 59 18 0 37 13 34 32
2 57 66 5 19 63 19 2

Resultant Matrix C:
8899 24808 21787 12969 20273 16075 20083 18868
7114 16734 16761 12228 15384 9556 16776 18112
6706 30020 25270 11894 19376 14453 19778 18204
11667 23963 24435 18894 24292 11201 20008 21433
5022 13936 12965 8792 13974 6499 12176 11853
8855 20756 19739 14114 20911 10768 17944 17632
12687 31308 28224 17482 24907 18381 22666 23449
12725 36024 32790 19264 30005 21755 27319 25083

Execution time = 0.000002 seconds
```

## Parallel

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <omp.h>

#define ROWS_A 8
#define COLS_A 8
#define ROWS_B 8
#define COLS_B 8

int main() {
    srand(time(NULL));

    double start_time, end_time;

    int A[ROWS_A][COLS_A];
    int B[ROWS_B][COLS_B];
    int C[ROWS_A][COLS_B];

    // Initialize matrices A and B with random values
    for (int i = 0; i < ROWS_A; i++) {
        for (int j = 0; j < COLS_A; j++) {
            // Random values between 0 and 99
            A[i][j] = rand() % 100;
        }
    }

    for (int i = 0; i < ROWS_B; i++) {
        for (int j = 0; j < COLS_B; j++) {
            // Random values between 0 and 99
            B[i][j] = rand() % 100;
        }
    }

    start_time = omp_get_wtime();

    omp_set_num_threads(8);

    // Perform matrix multiplication
    for (int i = 0; i < ROWS_A; i++) {
        for (int j = 0; j < COLS_B; j++) {
            C[i][j] = 0;
```

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

end_time = omp_get_wtime();

// Print the result matrix C
printf("Matrix A:\n");
for (int i = 0; i < ROWS_A; i++) {
    for (int j = 0; j < COLS_A; j++) {
        printf("%d ", A[i][j]);
    }
    printf("\n");
}

printf("\nMatrix B:\n");
for (int i = 0; i < ROWS_B; i++) {
    for (int j = 0; j < COLS_B; j++) {
        printf("%d ", B[i][j]);
    }
    printf("\n");
}

printf("\nResultant Matrix C:\n");
for (int i = 0; i < ROWS_A; i++) {
    for (int j = 0; j < COLS_B; j++) {
        printf("%d ", C[i][j]);
    }
    printf("\n");
}

double exec_time = end_time - start_time;
printf("\nExecution time = %lf seconds\n", exec_time);

return 0;
}
```

### Screenshots:

```
C:\Users\khush\Desktop\acads\7th sem\hpc1\p5>a.exe
```

```
Matrix A:
```

```
53 56 97 93 3 53 92 58
90 37 78 50 55 74 95 1
30 3 48 60 17 73 12 37
4 33 1 65 48 44 55 45
48 39 52 15 25 74 29 6
5 55 91 25 54 47 70 25
40 73 46 88 64 14 22 52
22 95 2 67 18 17 0 81
```

```
Matrix B:
```

```
14 90 51 92 24 21 59 58
1 67 99 76 9 66 47 30
5 3 80 47 25 55 70 82
79 32 58 45 71 61 69 49
77 29 59 19 68 7 35 77
36 29 29 8 34 73 57 52
43 5 68 4 14 94 20 92
38 74 69 60 91 51 17 3
```

```
Resultant Matrix C:
```

```
16929 18165 33373 22205 19376 31313 24918 28890
16659 16703 29313 19085 15670 26440 24019 32002
11262 10373 15636 11119 13380 15591 15036 15026
14584 10927 18274 10032 14658 17295 12395 15668
8220 11029 17346 12042 9036 15966 15115 17412
12365 10337 25464 13224 13190 21929 17369 23744
16169 17665 27317 19886 18883 19746 19445 20430
10782 17504 21717 17691 15363 16427 13502 10086
```

```
Execution time = 0.000000 seconds
```