Practical 2

PRN: 23520005
Name: Manas Indrapal Gedam
Batch: B7

GitHub Link: Link

Study and implementation of basic OpenMP clauses

Clauses:
num_threads(n) – Specify number of threads
parallel for – Automatically split loop iterations
private(var) – Each thread gets its own copy
shared(var) – All threads share one copy
reduction(+:var) – Safely combine results from all threads
schedule(static/dynamic) – Control how loop iterations are assigned to threads

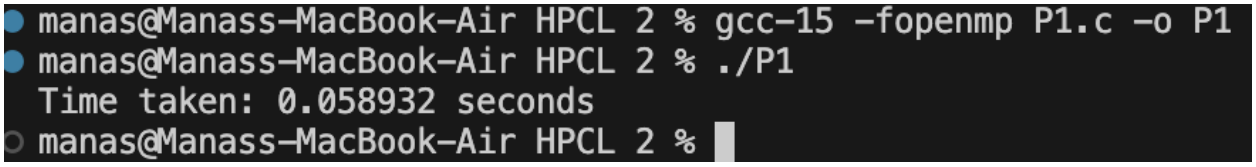
Problem Statement 1: Vector-Scalar Addition

Program:

P1.c > main()

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

int main() {
    int n = 100000000;
    float scalar = 2.5;
    float *A = malloc(n * sizeof(float));

    for (int i = 0; i < n; i++) {
        A[i] = i * 1.0;
    }

    double start = omp_get_wtime();

    #pragma omp parallel for num_threads(4)
    for (int i = 0; i < n; i++) {
        A[i] += scalar;
    }

    double end = omp_get_wtime();

    printf("Time taken: %f seconds\n", end - start);

    free(A);
    return 0;
}

```

Output:



Information

- Clause used: #pragma omp parallel for num_threads(n)
- Threads tested: 1, 2, 4, 8
- Data sizes tested: 1M, 10M, 100M elements

Analysis:
Execution time decreases as threads increase.
Performance gain slows down due to thread overhead beyond 4–8 threads.

Problem Statement 2: Calculation of Pi

Program:

P2.c > main()

```c
#include <stdio.h>
#include <omp.h>

int main() {
    int num_steps = 100000000;
    double step = 1.0 / (double)num_steps;
    double sum = 0.0;

    double start = omp_get_wtime();

    #pragma omp parallel for reduction(+:sum) num_threads(4)
    for (int i = 0; i < num_steps; i++) {
        double x = (i + 0.5) * step;
        sum += 4.0 / (1.0 + x * x);
    }

    double pi = sum * step;
    double end = omp_get_wtime();

    printf("Calculated Pi: %.15f\n", pi);
    printf("Time taken: %f seconds\n", end - start);
    return 0;
}
```

Output:

```
manas@Manass-MacBook-Air HPCL 2 % gcc-15 -fopenmp P2.c -o P2
manas@Manass-MacBook-Air HPCL 2 % ./P2
Calculated Pi: 3.141592653589683
Time taken: 0.131581 seconds
manas@Manass-MacBook-Air HPCL 2 %
```

Information

- Clause used: reduction(+:sum), num_threads(n)
- Threads tested: 1, 2, 4, 8

- Step count: 10M, 100M, 1B

Analysis:
The reduction clause ensures thread-safe summation.
Result is precise and consistent across all thread counts.