Shared Memory Programming with OpenMP
Programming Assignment – 6

Exercise – 1 [2 points] [OpenMP Implementation]

Suppose we toss darts randomly at a square dartboard, whose bullseye is at the origin, and whose sides are 0.6m in length. Suppose also that there's a circle inscribed in the square dartboard. The radius of the circle is 0.3m, and it's area is 0.093π square meters. If the points that are hit by the darts are uniformly distributed (and we always hit the square), then the number of darts that hit inside the circle should approximately satisfy the equation:

$$\text{number in circle} / \text{total number of tosses} = \pi/4$$

since the ratio of the area of the circle of the square is π/4. We can use this formula to estimate the value π with a random number generator:

```
number in circle = 0;
for (toss = 0; toss < number_of_tosses; toss++) {
x = random double between −1 and 1;
y = random double between −1 and 1;
distance_squared = x * x + y * y;
if (distance_squared <= 1) number_in_circle++;
}
pi_estimate = 4 * number_in_circle/((double) number_of_tosses);
```

Write a OpenMP program that uses a Monte Carlo method to estimate π. The main thread should read in the total number of tosses and print the estimate. You may want to use **long long int** for the number of hits in the circle and the number of tosses, since both may have to be very large to get a reasonable estimate of π.

Exercise – 2 [4 points]

Modify the trapezoidal rule program that uses a `parallel` for directive (`omp trap 3.c`, which will be given to you during the lectures) so that the `parallel` for is modified by a `schedule (runtime)` clause. Run the program with various assignments to the environment variable `OMP_SCHEDULE` and determine which iterations are assigned to which thread. This can be done by allocating an array `iterations` of `n` ints and in the `Trap` function assigning `omp_get_thread_num()` to `iterations[i]` in the i^{th} iteration of the for loop. What is the default assignment of iterations on your system? How are `guided` schedules determined?

Exercise – 3 [4 points]

In this task you will work again on the heat problem introduced in Assigment 3 (Exercise -3). The goal of this task is to take the given sequential code and produce multiple parallel versions using OpenMP. After testing the code for its correctness, you will collect timings to study the attained speedups, using 1, 2, 4, and 8 threads. You have to code the following 4 versions of the function `compute_*`.

a) Parallelize the outermost loop in the matrix update (lines 63-68).

b) Parallelize the innermost loop in the matrix update (lines 63-68).

- Which of these two versions do you expect to be faster? Why?

- Run the code and look at the timings. Was your prediction correct/accurate?

c) Take the fastest of the two version above and parallelize also the computation of the convergence. Use two separate parallel regions

d) Rewrite version "(c)" using once single parallel region

Create the 4 different functions (`compute_outer`, `compute_inner`, `compute_conv` and `compute_single_region`, respectively), and add the necessary code to the `main` function to evaluate the timings and speedups. Run experiments for $n \in [100, 500, 1000]$ using 1, 2, 4, and 8 threads.