A4 (55 marks)

***Focus**: OpenMP parallel for, loop-carried dependencies*

Q1. [10 marks] Repeat question Q1 from Assignment A3 using **OpenMP parallel for** loop(s). Check the output image to make sure your algorithm works correctly.
Would it be better to use nested parallel for loops (i.e., to parallelize both *for x* and *for y* loops)? Test using your code and explain.
Marking guide: +7 for code, +3 for the question about nested parallelism

Q2. [10 marks] Write a C program to perform parallel matrix multiplication using OpenMP. You should first create three matrices A, B, and C then initialize A and B to some values of your choice (e.g., a[i][j]=i+j and b[i][j]=i*j+1). Then, perform parallel matrix multiplication, and finally print out the result matrix C. In your code, try to improve the performance by (re)using the **same set of threads** for initializing A and B and for calculating C.
Search online for information about how to do matrix multiplication, e.g., here.
Include the following C statements near the beginning of your program (you may change the numbers).

```
#define NRA 20  /* number of rows in A */
#define NCA 30  /* number of columns in A = number of rows in B */
#define NCB 10  /* number of columns in matrix B */
```

Marking guide: +6 for correct code, +4 for proper parallelization (up to 4 marks may be deducted for not 'reusing' the same threads)

Q3. [10 marks] (*Exercise 5.8, p. 264 in PACHECO-11*): Consider the loop

```
a[0] = 0;
for(i=1; i<n; i++) a[i] = a[i-1] + i;
```

There's clearly a loop-carried dependence, as the value of a[i] can't be computed without the value of a[i?1]. Can you see a way to eliminate this dependence and parallelize the loop? ***Justify your answer.***

Q4. [10 marks] (Q 5.2, p. 267 in PACHECO-11): Suppose we toss darts randomly at a square dartboard, whose bullseye is at the origin, and whose sides are 2 feet in length. Suppose also that there's a circle inscribed in the square dartboard. The radius of the circle is 1 foot, and it's area is π square feet. 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

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

since the ratio of the area of the circle to the area of the square is $\pi/4$.

We can use this formula to estimate the value of 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);
```

This is called a "Monte Carlo" method, since it uses randomness (the dart tosses).

Write an OpenMP program that uses a Monte Carlo method to estimate π. Read in the total number of tosses before forking any threads. Use a `reduction` clause to find the total number of darts hitting inside the circle. Print the result after joining all the threads. You may want to use `long long int`s 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.

Note: Use the functions in the two files my_rand.c and my_rand.h to generate a random number between -1 and 1 given a seed value (the seed value could be your thread number, the current time, etc. Note that there are two functions available in these two files: one for generating a random integer and the other for a random double in the range [0,1]. You need to read the functions' description and then call the correct function. Then, you need to come up with a mathematical formula that uses the generated random number from 0 to 1 in order to compute a random number from -1 to 1 (i.e., map the range [0,1] to the range[-1,1]).
Marking guide: +5 for correct code, +5 for proper parallelization

Q5. [15 marks] (PACHECO-11 Q5.3(a,b,d) p. 268): Count sort is a simple serial sorting algorithm that can be implemented as follows:

```
void count_sort(int a[], int n) {
    int i, j, count;
    int* temp = malloc(n * sizeof(int));
    for (i = 0; i < n; i++){
        count = 0;
        for (j = 0; j < n; j++)
            if (a[j] < a[i])
                count++;
            else if (a[j] == a[i] && j < i)
                count++;
        temp[count] = a[i];
    }
    memcpy(a, temp, n * sizeof(int));
    free(temp);
}
```

The basic idea is that for each element `a[i]` in the list `a`, we count the number of elements in the list that are less than `a[i]`. Then we insert `a[i]` into a temporary list using the subscript determined by the count. There's a slight problem with this approach when the list contains equal elements, since they could get assigned to the same slot in the temporary list. The code deals with this by incrementing the count for equal elements on the basis of the subscripts. If both `a[i]==a[j]` and `j<i`, then we count `a[j]` as being "less than" `a[i]`. After the algorithm has completed, we overwrite the original array with the temporary array using the `memcpy`.

a) If we try to parallelize the `for i` loop (the outer loop), which variables should be private and which should be shared?

b) If we parallelize the `for i` loop using the scoping you specified in the previous part, are there any loop-carried dependences? Explain your answer.

c) Write a C program that includes a parallel implementation of `count_sort`.

Report the wall-clock time for both serial and parallel implementation with n=10000 and 8 threads.
Marking guide: +2 for (a), +3 for (b), and +10 for (d). Up to 3 marks are deducted if the time is not calculated and reported.

Submission Instructions

For this assignment, you need to do the following:

1- *Programming questions*: Create one C file for *each* programming question and write your answer inside that file. Your files should have the same name as the question number (e.g., Q1.c)

2- Non-programming questions:
   - If there are any discussion/essay questions related to a programming question, write your answers as comments at the end of your code for that question.
   - For all other non-programming questions (i.e., not related to any programming question), write your answers to all of them in *one* Word document file,

3- After solving all questions, compress all your files into one zip folder and give a name to the zipped file that matches your ID (e.g., 1234567.zip).

4- Submit the zipped file to Blackboard Connect.

Note that you can resubmit an assignment, but the new submission overwrites the old submission and receives a new timestamp.