Lab 3 – Report

Name: Deepa Subray Hegde
PSU ID: 954680127

Exercise 1:

1. Take a look at the program. How many copies of "Hello world!" do you think the program will print when executed? Compile and run the program. Is your guess right?
   linux> make hello-omp
   linux> ./hello-omp

   There are 8 cores in the linuxlab system. Therefore by default it creates 8 threads so print statement executes 8 times

2. Try to change the number of "Hello world!" printouts through two difference channels: (1) environment variables, (2) in-program directives. Run the program again to verify. If these two channels give conflicting directives, e.g. one says 4 and the other says 6, find out which one prevails

   Done

3. Modify the print statement so that you can see the thread and CPU core info with each message

   Done

Exercise 2:

1. The two OpenMP programs in loop-omp1.c and loop-omp2.c are attempts at parallelizing each of the two loops in the double loop-nest. Compile these programs and run each multiple times. Do they always work? If not, figure out the problem and fix it

   No. This is because when there is nested for loops, only the outer loop will be parallelized. Using collapse directive will parallelize both the loops. Also the total calculation statement should be put under the critical section by using critical directive.

2. Write a third OpenMP program in loop-omp3.c to parallelize both loops in the double loop-nest

Done

Exercise 3:

1. Two OpenMP programs, rec-omp1.c and rec-omp2.c, try to parallelize the recursive function. Try to compile and run them. Do you see expected parallelism? If not, can you figure out why

   No. This is because section directive is used to parallelize. Section is static and next section executes only after the first section finishes. All the operations are run by a single thread

2. Write a third OpenMP program in rec-omp3.c to parallelize the recursive function so that full parallelism is realized

   Done. This is done using task derivative

Exercise 4:

Done

Exercise 5:

1. Compile and run the program. How many lines of output do you see? What do you think happened to each of the two parallel directives?

   Main thread creates 2 child threads and each of the child creates 2 more child threads. By default the nested parallelism is disabled hence the output does not result in 6 lines of output.

2. Now set the environment variable OMP_NESTED to true. Run the program again. Do you see a different output?

   By setting the environment variable we are enabling the nested parallelism and the output is as expected with 2 child threads, each creating 2 more resulting in total of 6 lines of output

Exercise 6:

1. Insert OpenMP directives to the code so that the deposit and withdraw actions will be concurrent. Try three different versions: (a) with the sections directive, (b) with the task directive, and (c) with the for directive. Compile and run them. Verify that the resulting program for each case behaves as expected

   Done

2. Is there a need for synchronization? How would you handle it? Implement your solution

   Yes. Adding the updation of total amount should be under critical section so that multiple threads do not update the value at the same time.

Exercise 7:

Done