Laboratory for Software Analytics and Pervasive Parallelism
Prof. Ali Jannesari

jannesar@iastate.edu

Concurrent Systems (Coms 527x, Spring 23)

Exercise 04

Please submit your solution until March 12th, 2023 11:59 pm to the Canvas. If the solution includes programming tasks, please send well-commented source codes organized in a tar or zip archive. Also please time and screenshot all results.**Solutions which are delivered late, do not earn points.**

Task 1

(10 points)
Using OpenMP directives parallelize the given serial program **pi.c** which calculates an approximation of $\pi(pi)$ using an n-point quadrature rule:

$$\pi = \int_0^1 \frac{4}{1+x^2} \tag{1}$$

Verify your parallel version generates the same value as the original serial version when run with a single OpenMP thread, and approximately the same value when run multithreaded. Note: Please use 2 threads and 10000000 iterations.

Task 2

(10 points) See the file fibonacci.c supplied with the exercise. It computes the fibonacci number using a recursiveapproach. Your task is to parallelize the code with OpenMP Task construct. Use appropriate constructsto parallelize it. Please explain your additions in the code with comments. Use 16 threads to parallelize.

Task 3

(10 Points)
The example below attempts to show use of the parallel for construct. However it will generate errors at compile time. Try to determine what is causing the error and correct the program. Please just write your answer in your solution sheet.

```
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>
#define N           50
#define CHUNKSIZE   5

int main (int argc, char *argv[]) {
  int i, chunk, tid;
  float a[N], b[N], c[N];

/* Some initializations */
  for (i=0; i < N; i++)
    a[i] = b[i] = i * 1.0;
  chunk = CHUNKSIZE;

#pragma omp parallel for      \
  shared(a,b,c,chunk)         \
  private(i,tid)              \
  schedule(static,chunk)
  {
  tid = omp_get_thread_num();
  for (i=0; i < N; i++)
    {
    c[i] = a[i] + b[i];
    printf("tid= %d i= %d c[i]= %f\n", tid, i, c[i]);
    }
  }
  return 0;
}
```

Task 4

(10 Points) See the file for dot product supplied. It calculates dot product. Parallelize it with OpenMPand compare the results with the serial code. Please explain your additions in the code with comments.Use 16 threads to parallelize.

```
#include <stdio.h>
#include <stdlib.h>
/* Define length of dot product vectors */
#define VECLEN 100
int main (int argc, char* argv[])
{
int i,len=VECLEN;
double *a, *b;
double sum;
printf("Starting omp_dotprod_serial\n");
/* Assign storage for dot product vectors */
a = (double*) malloc (len*sizeof(double));
b = (double*) malloc (len*sizeof(double));
/* Initialize dot product vectors */
for (i=0; i<len; i++)
{
a[i]=10.0;
b[i]=a[i];
}
/* Perform the dot product */
sum = 0.0;
for (i=0; i<len; i++)
{
sum += (a[i] * b[i]);
}
printf ("Done. Serial version: sum  =  %f \n", sum);
free (a);free (b);
}
```