ITCS 5145 Spring 16 test 1 code questions

Solutions

The code has been tested. There are alternative acceptable answers.

Qu 21. Write a C program that will compute **pi** according to the series:

$$\mathbf{pi = 4(1 - 1/3 + 1/5 - 1/7 + \ldots (-1)^n/(2n-1))}$$

with ***N*** terms of the series where ***N*** is a defined constant. Be careful to write the code so that it can be parallelized later. In particular, determine the sign of a term in the series without a reference to other terms in the series. A term in the series is positive if it is an odd numbered term, i.e. the first term (1), third term (1/5), fifth term (1/9) etc. A term is negative if it is an even numbered term, i.e. the second term (1/3), the fourth term (1/7), the sixth term (1/11) etc. It is not necessary to give the C include statements, but declare all variables that you use.

6 point

```
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>
#define N 10000000

int main (int argc, char *argv[])  {
        long int i;
        double pi;
        double sign = 1.0;
        double sum = 0.0;

        for (i = 0; i < N; i++) {
                if (i % 2 == 0) sign = 1.0; else sign = -1.0;
                sum += sign/(2*i+1);
        }
        pi = 4.0 * sum;
        printf("Pi = %f\n", pi);

        return(0);
}
```

Qu 22 Re-write the pi program in Qu 21 to be an OpenMP program. Use a reduction clause. Be careful to declare any variables that need to be private in a private clause. Set the number of threads to 4. You do not need to give the OpenMP include statement, but declare all variables that you use.

6 points

```
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>
#define N 10000000

int main (int argc, char *argv[])  {

        long int i;
        double pi;
        double sign = 1.0;
        double sum = 0.0;

        omp_set_num_threads(4);

        #pragma omp parallel for reduction (+:sum) private(sign)
        for (i = 0; i < N; i++) {
                if (i % 2 == 0) sign = 1.0; else sign = -1.0;
                sum += sign/(2*i+1);
        }
        pi = 4.0 * sum;
        printf("Pi = %f\n", pi);

        return(0);
}
```

Qu 23: Re-write the program in Qu 21 to be an MPI program. Each process is to use ***M*** terms in the series where ***M*** is a defined constant. The first process uses the first ***M*** terms in the series. The second process uses the next ***M*** terms the series. The third process uses the next ***M*** terms in the series etc. Use an MPI reduction routine to get the final answer. You do not need to give the MPI include statement, but declare all variables that you use.

8 points

```
#include "mpi.h"
#include <stdio.h>
#include <stdlib.h>
#define M 1000

int main (int argc, char *argv[])  {
        int size,rank;
        long int i,start,end;
        double pi,result;
        double sign = 1.0;
        double sum = 0.0;
        MPI_Status status;

        MPI_Init(&argc, &argv);
        //MPI_Comm_size(MPI_COMM_WORLD, &size);
        MPI_Comm_rank(MPI_COMM_WORLD, &rank);

        start = M * rank;
        end = start + M;
        for (i = start; i < end; i++) {
                if (i % 2 == 0) sign = 1.0; else sign = -1.0;
                sum += sign/(2*i+1);
        }

        MPI_Reduce (&sum, &result, 1, MPI_DOUBLE, MPI_SUM, 0, MPI_COMM_WORLD);

        if (rank == 0) {
                pi = 4.0 * result;
                printf("Pi = %f\n", pi);
        }

        MPI_Finalize();

        return(0);
}
```