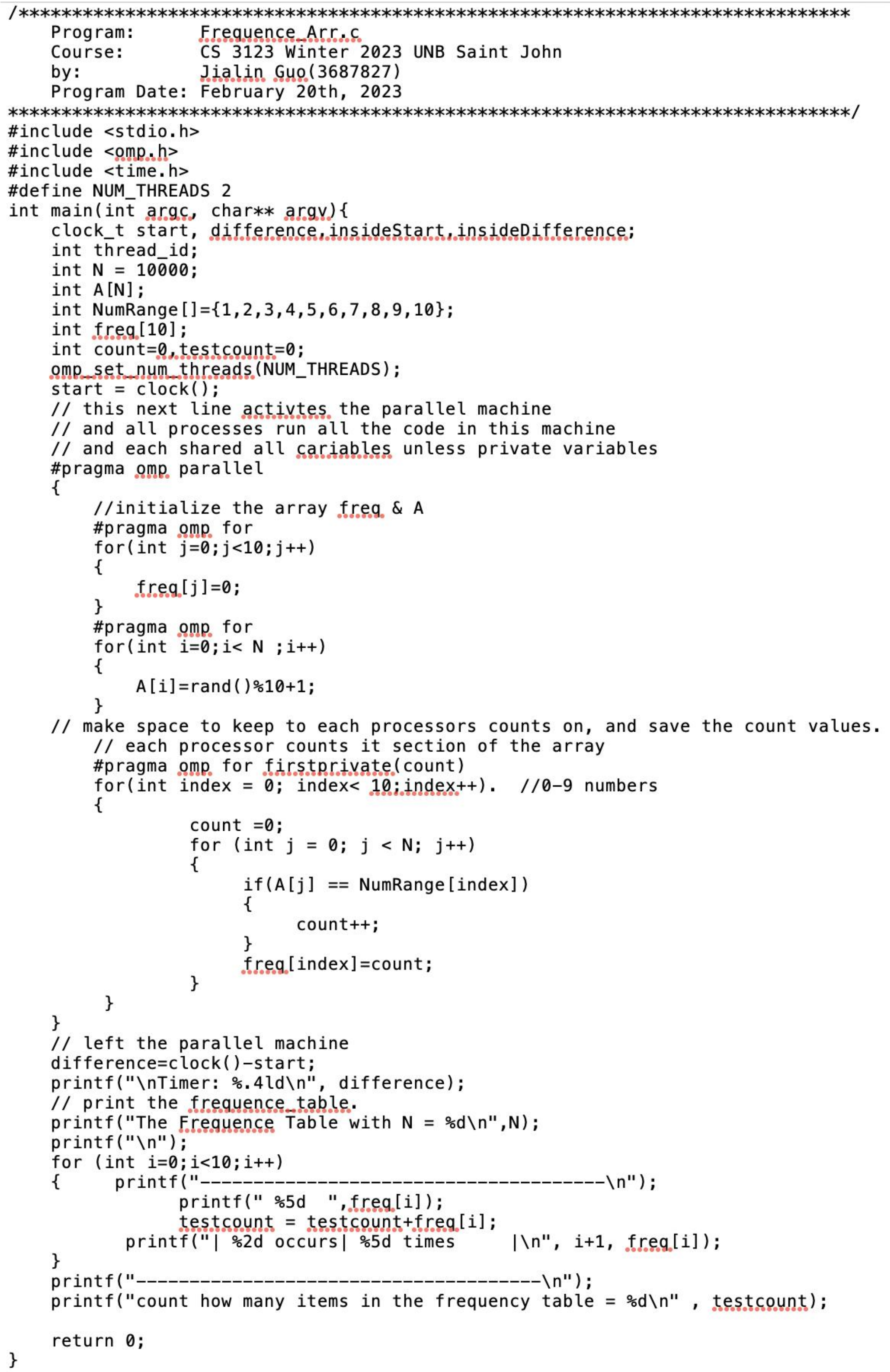
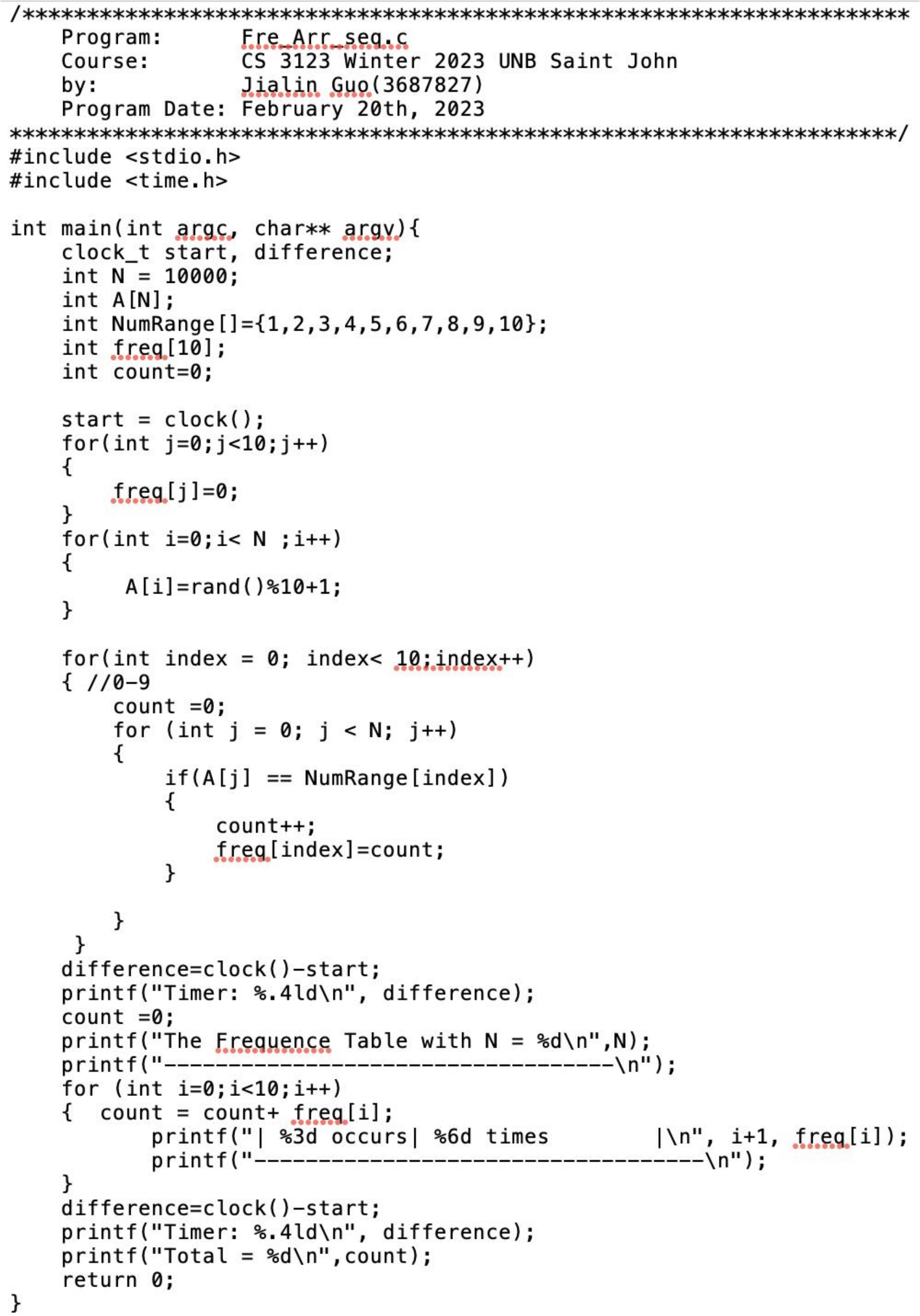
CS 3123 (McNally) : High Speed Computing Winter 2023
Jialin Guo #3687827

Q1 pragma omp parallel



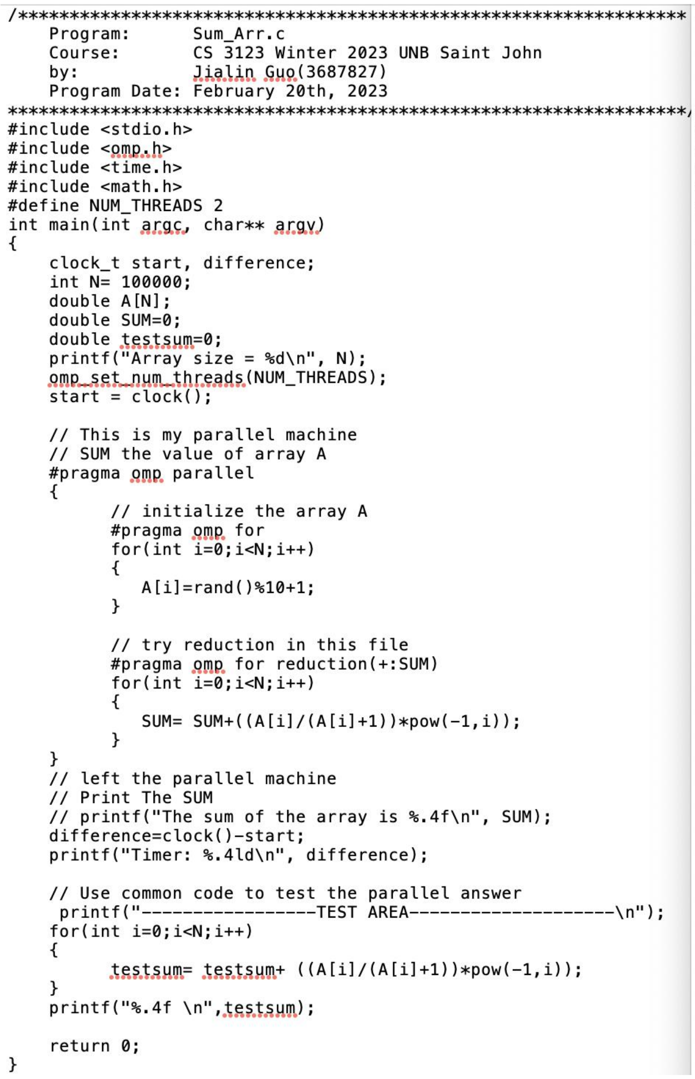
```
/*******************************************************************************
    Program:      Frequence_Arr.c
    Course:       CS 3123 Winter 2023 UNB Saint John
    by:           Jialin Guo(3687827)
    Program Date: February 20th, 2023
*******************************************************************************/
#include <stdio.h>
#include <omp.h>
#include <time.h>
#define NUM_THREADS 2
int main(int argc, char** argv){
    clock_t start, difference,insideStart,insideDifference;
    int thread_id;
    int N = 10000;
    int A[N];
    int NumRange[]={1,2,3,4,5,6,7,8,9,10};
    int freq[10];
    int count=0,testcount=0;
    omp_set_num_threads(NUM_THREADS);
    start = clock();
    // this next line activtes the parallel machine
    // and all processes run all the code in this machine
    // and each shared all cariables unless private variables
    #pragma omp parallel
    {
        //initialize the array freq & A
        #pragma omp for
        for(int j=0;j<10;j++)
        {
            freq[j]=0;
        }
        #pragma omp for
        for(int i=0;i< N ;i++)
        {
            A[i]=rand()%10+1;
        }
    // make space to keep to each processors counts on, and save the count values.
        // each processor counts it section of the array
        #pragma omp for firstprivate(count)
        for(int index = 0; index< 10;index++).  //0-9 numbers
        {
                 count =0;
                 for (int j = 0; j < N; j++)
                 {
                      if(A[j] == NumRange[index])
                      {
                           count++;
                      }
                      freq[index]=count;
                 }
         }
    }
    // left the parallel machine
    difference=clock()-start;
    printf("\nTimer: %.4ld\n", difference);
    // print the frequence_table.
    printf("The Frequence Table with N = %d\n",N);
    printf("\n");
    for (int i=0;i<10;i++)
    {     printf("--------------------------------------\n");
                printf(" %5d  ",freq[i]);
                testcount = testcount+freq[i];
           printf("| %2d occurs| %5d times     |\n", i+1, freq[i]);
    }
    printf("--------------------------------------\n");
    printf("count how many items in the frequency table = %d\n" , testcount);

    return 0;
}
```

Q1 sequential

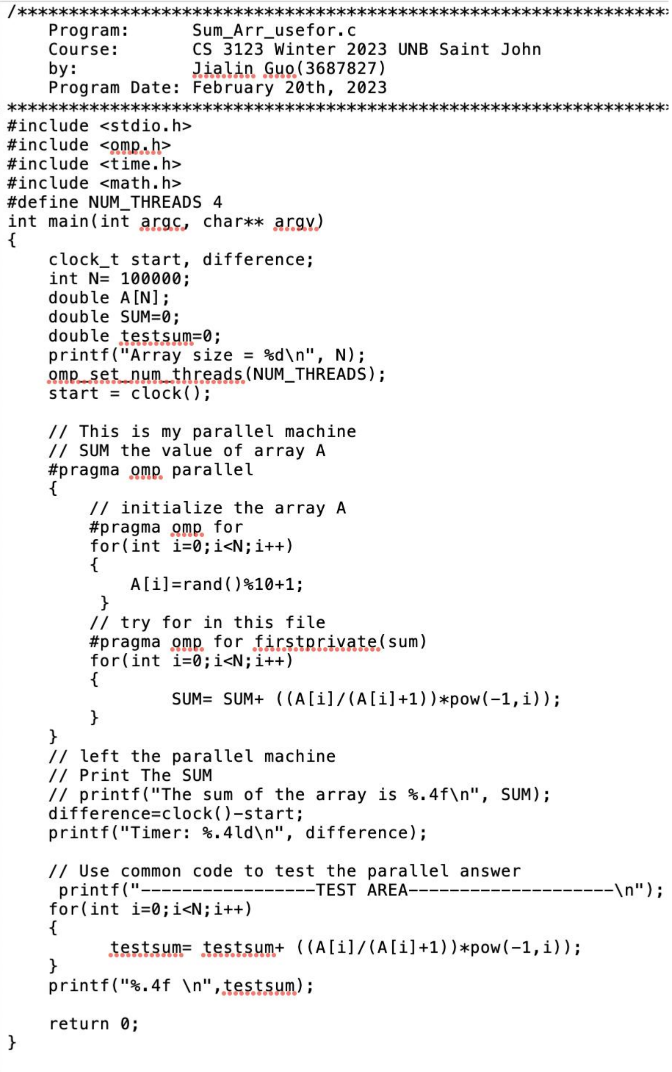


```
/**********************************************************************
    Program:      Fre_Arr_seq.c
    Course:       CS 3123 Winter 2023 UNB Saint John
    by:           Jialin Guo(3687827)
    Program Date: February 20th, 2023
**********************************************************************/
#include <stdio.h>
#include <time.h>

int main(int argc, char** argv){
    clock_t start, difference;
    int N = 10000;
    int A[N];
    int NumRange[]={1,2,3,4,5,6,7,8,9,10};
    int freq[10];
    int count=0;

    start = clock();
    for(int j=0;j<10;j++)
    {
        freq[j]=0;
    }
    for(int i=0;i< N ;i++)
    {
         A[i]=rand()%10+1;
    }

    for(int index = 0; index< 10;index++)
    { //0-9
        count =0;
        for (int j = 0; j < N; j++)
        {
            if(A[j] == NumRange[index])
            {
                count++;
                freq[index]=count;
            }

        }
     }
    difference=clock()-start;
    printf("Timer: %.4ld\n", difference);
    count =0;
    printf("The Frequence Table with N = %d\n",N);
    printf("-----------------------------------\n");
    for (int i=0;i<10;i++)
    {  count = count+ freq[i];
           printf("| %3d occurs| %6d times        |\n", i+1, freq[i]);
           printf("-----------------------------------\n");
    }
    difference=clock()-start;
    printf("Timer: %.4ld\n", difference);
    printf("Total = %d\n",count);
    return 0;
}
```

Q2 pragma omp parallel

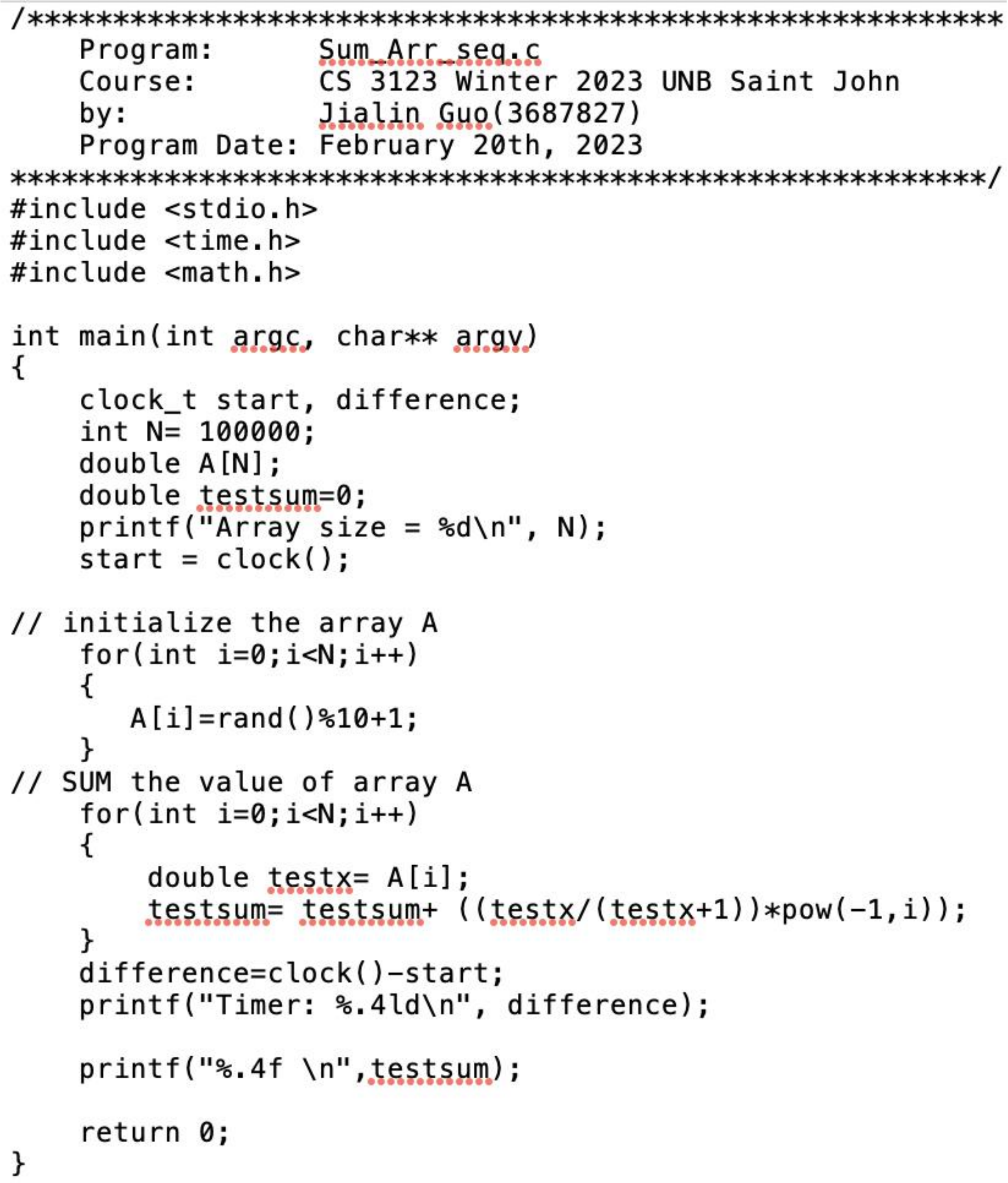


```
/*****************************************************************
    Program:      Sum_Arr.c
    Course:       CS 3123 Winter 2023 UNB Saint John
    by:           Jialin Guo(3687827)
    Program Date: February 20th, 2023
*****************************************************************/
#include <stdio.h>
#include <omp.h>
#include <time.h>
#include <math.h>
#define NUM_THREADS 2
int main(int argc, char** argv)
{
    clock_t start, difference;
    int N= 100000;
    double A[N];
    double SUM=0;
    double testsum=0;
    printf("Array size = %d\n", N);
    omp_set_num_threads(NUM_THREADS);
    start = clock();

    // This is my parallel machine
    // SUM the value of array A
    #pragma omp parallel
    {
          // initialize the array A
          #pragma omp for
          for(int i=0;i<N;i++)
          {
             A[i]=rand()%10+1;
          }

          // try reduction in this file
          #pragma omp for reduction(+:SUM)
          for(int i=0;i<N;i++)
          {
             SUM= SUM+((A[i]/(A[i]+1))*pow(-1,i));
          }
    }
    // left the parallel machine
    // Print The SUM
    // printf("The sum of the array is %.4f\n", SUM);
    difference=clock()-start;
    printf("Timer: %.4ld\n", difference);

    // Use common code to test the parallel answer
     printf("-----------------TEST AREA--------------------\n");
    for(int i=0;i<N;i++)
    {
          testsum= testsum+ ((A[i]/(A[i]+1))*pow(-1,i));
    }
    printf("%.4f \n",testsum);

    return 0;
}
```



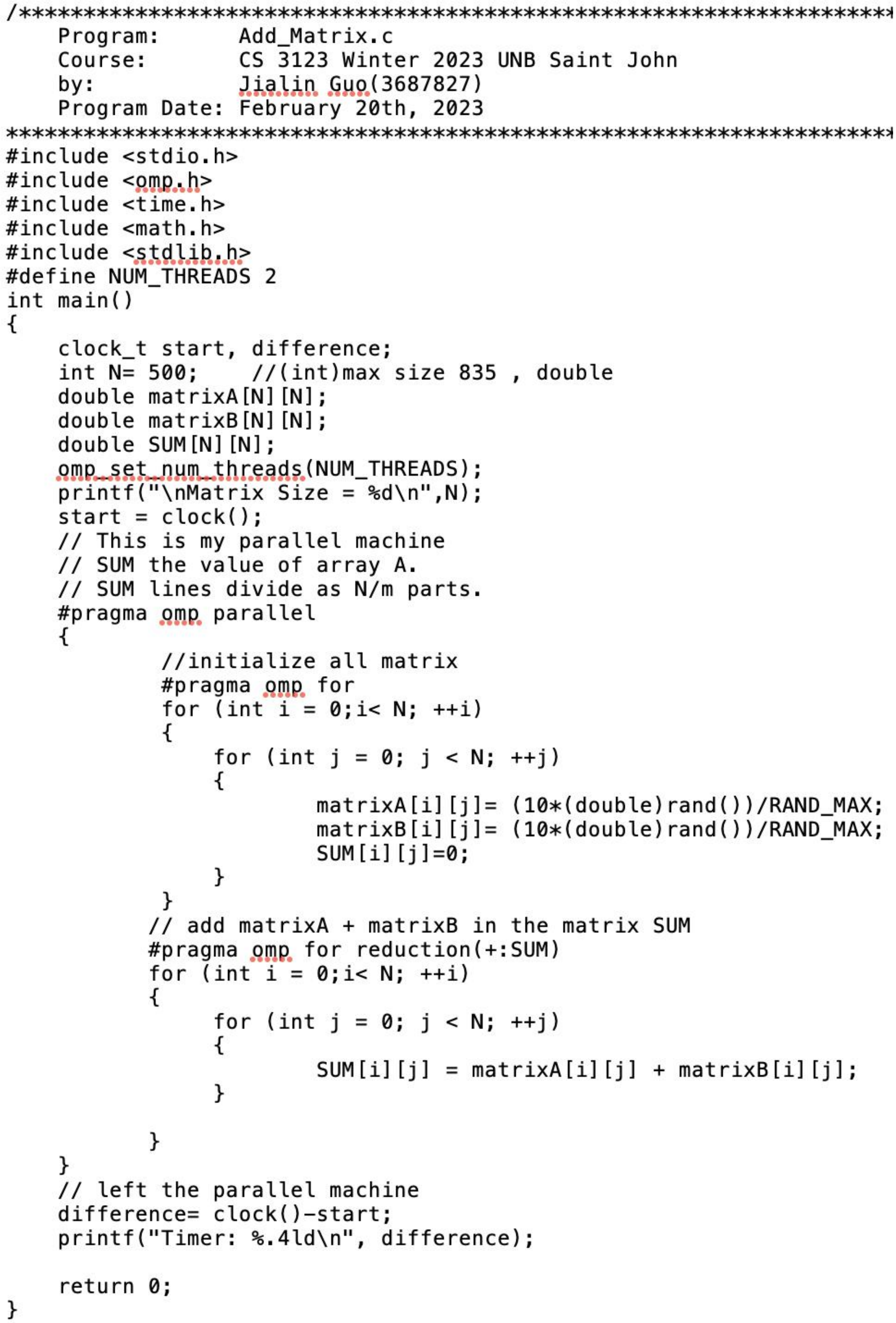
```
/****************************************************************
    Program:      Sum_Arr_usefor.c
    Course:       CS 3123 Winter 2023 UNB Saint John
    by:           Jialin Guo(3687827)
    Program Date: February 20th, 2023
****************************************************************
#include <stdio.h>
#include <omp.h>
#include <time.h>
#include <math.h>
#define NUM_THREADS 4
int main(int argc, char** argv)
{
    clock_t start, difference;
    int N= 100000;
    double A[N];
    double SUM=0;
    double testsum=0;
    printf("Array size = %d\n", N);
    omp_set_num_threads(NUM_THREADS);
    start = clock();

    // This is my parallel machine
    // SUM the value of array A
    #pragma omp parallel
    {
        // initialize the array A
        #pragma omp for
        for(int i=0;i<N;i++)
        {
            A[i]=rand()%10+1;
         }
        // try for in this file
        #pragma omp for firstprivate(sum)
        for(int i=0;i<N;i++)
        {
                SUM= SUM+ ((A[i]/(A[i]+1))*pow(-1,i));
        }
    }
    // left the parallel machine
    // Print The SUM
    // printf("The sum of the array is %.4f\n", SUM);
    difference=clock()-start;
    printf("Timer: %.4ld\n", difference);

    // Use common code to test the parallel answer
     printf("-----------------TEST AREA--------------------\n");
    for(int i=0;i<N;i++)
    {
          testsum= testsum+ ((A[i]/(A[i]+1))*pow(-1,i));
    }
    printf("%.4f \n",testsum);

    return 0;
}
```

Q2 sequential



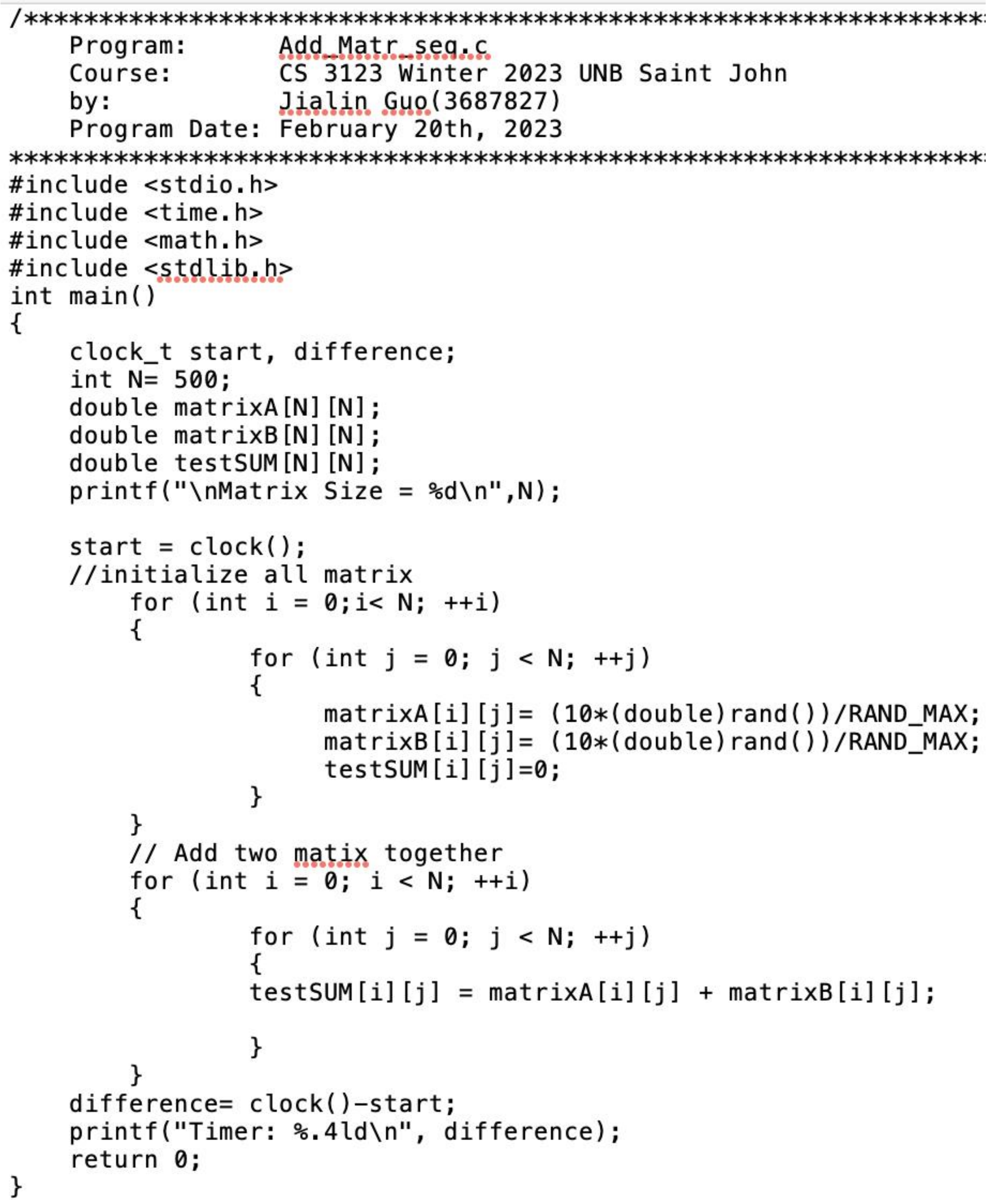
```
/*********************************************************
    Program:      Sum_Arr_seq.c
    Course:       CS 3123 Winter 2023 UNB Saint John
    by:           Jialin Guo(3687827)
    Program Date: February 20th, 2023
*********************************************************/
#include <stdio.h>
#include <time.h>
#include <math.h>

int main(int argc, char** argv)
{
    clock_t start, difference;
    int N= 100000;
    double A[N];
    double testsum=0;
    printf("Array size = %d\n", N);
    start = clock();

// initialize the array A
    for(int i=0;i<N;i++)
    {
       A[i]=rand()%10+1;
    }
// SUM the value of array A
    for(int i=0;i<N;i++)
    {
        double testx= A[i];
        testsum= testsum+ ((testx/(testx+1))*pow(-1,i));
    }
    difference=clock()-start;
    printf("Timer: %.4ld\n", difference);

    printf("%.4f \n",testsum);

    return 0;
}
```

Q3 pragma omp parallel



```
/*******************************************************************
    Program:      Add_Matrix.c
    Course:       CS 3123 Winter 2023 UNB Saint John
    by:           Jialin Guo(3687827)
    Program Date: February 20th, 2023
********************************************************************
#include <stdio.h>
#include <omp.h>
#include <time.h>
#include <math.h>
#include <stdlib.h>
#define NUM_THREADS 2
int main()
{
    clock_t start, difference;
    int N= 500;    //(int)max size 835 , double
    double matrixA[N][N];
    double matrixB[N][N];
    double SUM[N][N];
    omp_set_num_threads(NUM_THREADS);
    printf("\nMatrix Size = %d\n",N);
    start = clock();
    // This is my parallel machine
    // SUM the value of array A.
    // SUM lines divide as N/m parts.
    #pragma omp parallel
    {
            //initialize all matrix
            #pragma omp for
            for (int i = 0;i< N; ++i)
            {
                for (int j = 0; j < N; ++j)
                {
                        matrixA[i][j]= (10*(double)rand())/RAND_MAX;
                        matrixB[i][j]= (10*(double)rand())/RAND_MAX;
                        SUM[i][j]=0;
                }
            }
           // add matrixA + matrixB in the matrix SUM
           #pragma omp for reduction(+:SUM)
           for (int i = 0;i< N; ++i)
           {
                for (int j = 0; j < N; ++j)
                {
                        SUM[i][j] = matrixA[i][j] + matrixB[i][j];
                }

           }
    }
    // left the parallel machine
    difference= clock()-start;
    printf("Timer: %.4ld\n", difference);

    return 0;
}
```

Q3 sequential



```
/*****************************************************************
    Program:      Add_Matr_seq.c
    Course:       CS 3123 Winter 2023 UNB Saint John
    by:           Jialin Guo(3687827)
    Program Date: February 20th, 2023
******************************************************************
#include <stdio.h>
#include <time.h>
#include <math.h>
#include <stdlib.h>
int main()
{
    clock_t start, difference;
    int N= 500;
    double matrixA[N][N];
    double matrixB[N][N];
    double testSUM[N][N];
    printf("\nMatrix Size = %d\n",N);

    start = clock();
    //initialize all matrix
        for (int i = 0;i< N; ++i)
        {
                for (int j = 0; j < N; ++j)
                {
                     matrixA[i][j]= (10*(double)rand())/RAND_MAX;
                     matrixB[i][j]= (10*(double)rand())/RAND_MAX;
                     testSUM[i][j]=0;
                }
        }
        // Add two matix together
        for (int i = 0; i < N; ++i)
        {
                for (int j = 0; j < N; ++j)
                {
                testSUM[i][j] = matrixA[i][j] + matrixB[i][j];

                }
        }
    difference= clock()-start;
    printf("Timer: %.4ld\n", difference);
    return 0;
}
```

1. Create a table of times T(m) and Speedups S(m) for M = 1, 2, 3, and 4 processors. Treat either your sequential time or the M = 1 time as the sequential time - T(1) - they should be similar. Use the same data collection method for this week as last week (use last weeks data too of course) (different day has different running time.)

	T(1)	T(2)	S(2)	T(3)	S(3)	T(4)	S(4)
Q1 from Asgn 2	1491	5336	0.279422789	10191	0.146305564	12156	0.122655479
Q2 from Asgn 2	3996	18083	0.220981032	32299	0.123719001	38250	0.104470588
Q3 from Asgn 2	3682	79068	0.046567512	139700	0.026356478	149844	0.024572222

and

	T(1)	T(2)	S(2)	T(3)	S(3)	T(4)	S(4)
Q1 from Asgn 3	1409	5704	0.247019635	10374	0.13582032	13082	0.107705244
Q2 from Asgn 3	4034	19062	0.211625223	33317	0.121079329	40340	0.1
Q3 from Asgn 3	3650	71848	0.050801692	137876	0.026473063	154307	0.023654144

2. Efficiency is the ratio of speed up to the number of processors. We can view it as the percentage of time onaverage that the processors are in use. Create a table for efficiency in the same way as above.

Question	S(P)	p	E(p)	E(P)*100%
1Freq	0.107705244	4	0.026926311	2.6926311
2Array	0.1	4	0.025	2.5
3Matrix	0.023654144	4	0.005913536	0.5913536
1Freq	0.13582032	3	0.04527344	4.527344
2Array	0.121079329	3	0.040359776	4.035977633
3Matrix	0.026473063	3	0.008824354	0.882435433
1Freq	0.247019635	2	0.123509818	12.35098175
2Array	0.211625223	2	0.105812612	10.58126115
3Matrix	0.050801692	2	0.025400846	2.5400846