NAME – ANSH GOEL
REGISTRATION NO.- 20BCE1798

VIT UNIVERSITY
(Estd. u/s 3 of UGC Act 1956)
www.vit.ac.in
Vellore ▪ Chennai
CHENNAI CAMPUS
Vandalur - Kelambakkam Road, Chennai - 600127

Programme	:	**B.Tech.(CSE)**	Semester	:	**Fall '22-23**
Course	:	**Parallel and Distributed Computing**	Code	:	**CSE4001**
Faculty	:	**R. Kumar**	Slot	:	**L9+L10**

1. Write a program in OpenMP to find out the largest number in an array of 1000000 randomly generated numbers from 1 to 100000 using reduction clause. Compare the versions of serial, parallel for and reduction clause.

test.c > main()

```c
#include<stdio.h>
#include<omp.h>
#include<stdlib.h>
#include<limits.h>
#include<time.h>

int main()
{
    long long int A[1000000];
    for(int i=0;i<1000000;i++)
        A[i]=(rand()%100000)+1;
    int max=INT_MIN;
    clock_t t;
    t=clock();
    for(int i=0;i<1000000;i++)
    {
        if(A[i]>max)
        {
            max=A[i];

        }

    }

    t=clock()-t;

    printf("Serial method  answer: %d  with a time %f\n\n",max,(double)t/(double)CLOCKS_PER_SEC);
```

```
29
30      max=INT_MIN;
31
32
33
34      t=clock();
35      #pragma omp parallel for shared(max,A) schedule(auto)
36
37          for(int i=0;i<1000000;i++)
38          {
39              if(A[i]>max)
40              {
41                  max=A[i];
42
43              }
44
45          }
46      t=clock()-t;
47
48      printf("Parallel method  answer: %d with a time %f\n\n",max,(double)t/(double)CLOCKS_PER_SEC);
```

```
51      int max_val=INT_MIN;
52
53      t=clock();
54      #pragma omp parallel for shared(A) reduction(max:max_val)
55          for(int i=0;i<1000000;i++)
56          {
57              if(A[i]>max_val)
58              {
59                  max_val=A[i];
60
61              }
62
63          }
64      t=clock()-t;
65
66      printf("Parallel method with reduction answer: %d with a time %f\n\n",max_val,(double)t/(double)CLOCKS_PER_SEC);
67
68  }
```

```
Serial method  answer: 100000  with a time 0.003114

Parallel method  answer: 100000 with a time 0.002972

Parallel method with reduction answer: 100000 with a time 0.002903
```

2. Write a program in OpenMP to find out the standard deviation of 1000000 randomly generated numbers using reduction clause. Document the development versions of serial, parallel for and reduction clause.

```c
#include<stdio.h>
#include<omp.h>
#include<stdlib.h>
#include<math.h>
#define n 1000000

int main()
{
    int X[n];
    for(int i=0;i<n;i++)
        X[i]=(rand()%100)+1;

    double start =omp_get_wtime();
    //Serial approach
    double mean=0;
    for(int i=0;i<n;i++)
        mean+=X[i];
    mean=mean/n;
    double sd=0;
    for(int i=0;i<n;i++)
        sd+=pow(mean-X[i],2);
    sd=sd/n;
    sd=pow(sd,0.5);
    printf("Serial method answer is :%f and time is %f\n\n",sd,omp_get_wtime()-start);

    //parallel

    mean=0;
    start=omp_get_wtime();
    #pragma omp parallel for shared(X,mean) schedule(auto)
        for(int i=0;i<n;i++)
            mean+=X[i];
    mean=mean/n;
    sd=0;
    #pragma omp parallel for shared(X,sd) schedule(auto)
        for(int i=0;i<n;i++)
            sd+=pow(mean-X[i],2);
    sd=sd/n;
    sd=pow(sd,0.5);
    printf("Parallel method answer is :%f and time is %f\n\n",sd,omp_get_wtime()-start);

}
```

```
42      //Reduction
43      mean=0;
44      start=omp_get_wtime();
45      #pragma omp parallel for shared(X) reduction(+:mean) schedule(auto)
46          for(int i=0;i<n;i++)
47              mean+=X[i];
48      mean=mean/n;
49      sd=0;
50      #pragma omp parallel for shared(X) reduction(+:sd) schedule(auto)
51          for(int i=0;i<n;i++)
52              sd+=pow(mean-X[i],2);
53      sd=sd/n;
54      sd=pow(sd,0.5);
55      printf("Reduction method answer is :%f and time is %f",sd,omp_get_wtime()-start);
56
57  }
```

```
Serial method answer is :28.860555 and time is 0.198005

Parallel method answer is :28.860555 and time is 0.168805

Reduction method answer is :28.860555 and time is 0.142916sid@sid-VirtualBox:~/PDC/LAB6$
```

3. Write a multithreaded program using OpenMP to implement sequential and parallel version of the Monte Carlo algorithm for approximating Pi. Compare the results of sequential, loop-level parallelism and reduction clause with 10000000 samples.

Q3.c > main()

```c
#include<stdio.h>
#include <omp.h>
#include<stdlib.h>
#include<math.h>
#define INTERVAL 10000

void main()
{
    double interval;
    int i;
    double rand_x, rand_y, origin_dist, pi;
    int circle_points = 0, square_points = 0;
    //Serial
    double start=omp_get_wtime();
    for (i = 0; i < (INTERVAL * INTERVAL); i++) {


        rand_x = (double)(rand() % (INTERVAL + 1)) / INTERVAL;
        rand_y = (double)(rand() % (INTERVAL + 1)) / INTERVAL;
        origin_dist = pow(rand_x,2)+pow(rand_y,2);
        if (origin_dist <= 1)
            circle_points++;
        square_points++;


        pi = (double)(4 * circle_points) / square_points;

    }

    printf("Pi in Serial is %f int time %f\n\n",pi,omp_get_wtime()-start);
```

```
33      //Parallel
34      circle_points = 0, square_points = 0;
35      start=omp_get_wtime();
36      #pragma omp parallel for shared(circle_points,square_points)
37          for (i = 0; i < (INTERVAL * INTERVAL); i++) {
38
39
40              rand_x = (double)(rand() % (INTERVAL + 1)) / INTERVAL;
41              rand_y = (double)(rand() % (INTERVAL + 1)) / INTERVAL;
42              origin_dist = pow(rand_x,2)+pow(rand_y,2);
43              if (origin_dist <= 1)
44                  circle_points++;
45              square_points++;
46
47      }
48      pi = (double)(4 * circle_points) / square_points;
49
50
51      printf("Pi in Parallel is %f int time %f\n\n",pi,omp_get_wtime()-start);
```

```
53      //Reduction
54      circle_points = 0, square_points = 0;
55      start=omp_get_wtime();
56      #pragma omp parallel for reduction(+:circle_points,square_points)
57          for (i = 0; i < (INTERVAL * INTERVAL); i++) {
58
59
60              rand_x = (double)(rand() % (INTERVAL + 1)) / INTERVAL;
61              rand_y = (double)(rand() % (INTERVAL + 1)) / INTERVAL;
62              origin_dist = pow(rand_x,2)+pow(rand_y,2);
63              if (origin_dist <= 1)
64                  circle_points+=1;
65              square_points+=1;
66
67      }
68      pi = (double)(4 * circle_points) / square_points;
69
70
71      printf("Pi in Reduction is %f int time %f\n\n",pi,omp_get_wtime()-start);
72  }
```

```
Pi in Serial is 3.140928 int time 14.712609

Pi in Parallel is 3.141707 int time 14.254263

Pi in Reduction is 3.141629 int time 14.123284
```