**Program5**

**Write an OpenMP parallel program for Points Classification. Prove the correctness of sequential program with that of parallel**

#include<stdio.h>

#include<stdlib.h>

#include<omp.h>

#include<math.h>

#define k 4

#define POINTS\_SIZE 1000000

#define PRINT\_POINTS 0

double cur\_distance;

double cluster\_mean[k];

long long cluster\_count[k];

int points[POINTS\_SIZE][2];

void populate\_points()

{

long long i;

for(i=0;i<POINTS\_SIZE;i++)

{

srand(i);

points[i][0]=rand()%100;

points[i][1]=-1;

}

for(i=0;i<k;i++)

{

cluster\_mean[i]=points[i][0];

cluster\_count[i]=1;

points[i][1]=i;

}

}

double get\_distance(double x,double y){

double diff=x-y;

if(diff<0)

diff\*=-1;

return diff;

}

void main()

{

int nt,cluster\_index,j;

double t,min\_dist,cur\_dist;

long long i;

printf("\en enter the number of threads : ");

scanf("%d",&nt);

printf("\n");

populate\_points();

t=omp\_get\_wtime();

#pragma omp parallel for private(i,j,min\_dist,cur\_dist,cluster\_index) num\_threads(nt)

for(i=0;i<POINTS\_SIZE;i++)

{

min\_dist=1000,cur\_dist=-1;

cluster\_index=-1;

for(j=0;j<k;j++)

{

cur\_dist=get\_distance((double)points[i][0],cluster\_mean[j]);

if(cur\_dist<min\_dist)

{

min\_dist=cur\_dist;

cluster\_index=j;

}

}

if(PRINT\_POINTS!=0)

printf("\n%d belongs to cluster %d",points[i][0],cluster\_index+1);

points[i][1]=cluster\_index;

#pragma omp critical

{

cluster\_mean[cluster\_index]=((cluster\_mean[cluster\_index]\*cluster\_count[cluster\_index])

+points[i][0])/(cluster\_count[cluster\_index]+1);

cluster\_count[cluster\_index]++;

}

}

t=omp\_get\_wtime()-t;

printf("\n\nTime taken :%lf\n",t);

}



