Name: Anshul Agrawal

USN: 1RV17CS021

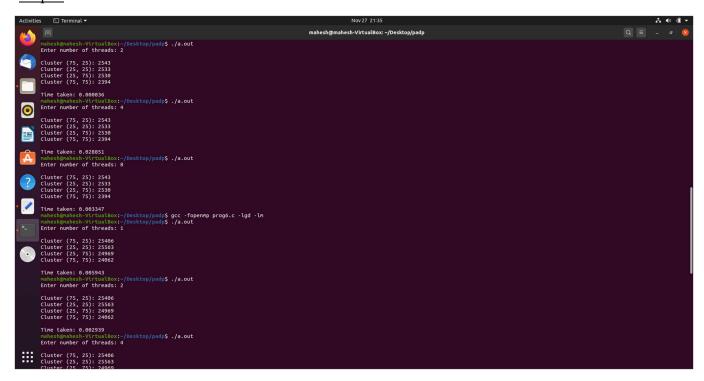
PADP lab Program 6

```
Code:
#include<stdio.h>
#include<omp.h>
#include<math.h>
#include<stdlib.h>
#define CLUSTER_SIZE 4
#define POINTS_SIZE 1000000
#define PRINT_POINTS 0
int cluster[CLUSTER_SIZE][2] = {{75, 25}, {25, 25}, {25, 75}, {75, 75}};
long long cluster_count[CLUSTER_SIZE];
int points[POINTS_SIZE][2];
void populate_points() {
  long long i;
for(i = 0; i < CLUSTER_SIZE; i++) {</pre>
                cluster_count[i] = 0;
        }
for(i = 0; i < POINTS_SIZE; i++) {</pre>
                      srand(i);
            points[i][0] = rand() % 100;
            points[i][1] = rand() % 100;
       }
}
```

```
double get_distance(int x1, int y1, int x2, int y2) {
int x = x2-x1, y = y2-y1;
return (double)sqrt((x * x) + (y * y));
}
int main() {
    double t;
populate_points();
  long long i;
if(PRINT_POINTS != 0) {
         for(i = 0; i < CLUSTER_SIZE; i++) {</pre>
                                  printf("\nCluster %lld : (%d, %d)", i+1, cluster[i][0], cluster[i][1]);
                          }
                   printf("\n\n");
        }
   int nt = 0;
printf("Enter number of threads: ");
scanf("%d", &nt);
t = omp_get_wtime();
#pragma omp parallel for private(i) shared(points, cluster) reduction(+:cluster_count) num_threads(nt)
for(i = 0; i < POINTS_SIZE; i++) {</pre>
        double min_dist = 100, cur_dist = -1;
              int j, cluster_index = -1;
```

```
for(j = 0; j < CLUSTER_SIZE; j++) {</pre>
                                   cur_dist = get_distance(points[i][0], points[i][1], cluster[j][0], cluster[j][1]);
                                   if(cur_dist<min_dist) {</pre>
                                            min_dist = cur_dist;
                                            cluster_index = j;
                                   }
                           }
               if(PRINT_POINTS != 0) {
                                   printf("\n(%d, %d) belongs to (%d, %d)", points[i][0], points[i][1],
cluster[cluster_index][0], cluster[cluster_index][1]);
                           }
          cluster_count[cluster_index]++;
        }
t = omp_get_wtime() - t;
for(i = 0; i < CLUSTER_SIZE; i++) {</pre>
         printf("\nCluster (%d, %d): %lld", cluster[i][0], cluster[i][1], cluster_count[i]);
        }
printf("\n\nTime taken: %If\n", t);
    return 0;
}
```

Output:



Graphs:

