NAME – GAUTAM CHANDRA SAHA
REG NO – 201900099
DATE – 23/02/2022

Q1. Implement quick sort parallel using open mp

```
// source code
#include <iostream>
#include <vector>
#include <omp.h>
using namespace std;
typedef vector<int> vi;
typedef vector<double> vd;

class Parallel {

   int partition(vi &arr, int start, int end){

      int pivot = arr[end];
      int i = (start - 1);

      for (int j = start; j <= end - 1; j++)
         if (arr[j] < pivot)
            swap(arr[++i], arr[j]);

      swap(arr[i + 1], arr[end]);

   return i + 1;
   }

 public:
   void quicksort(vi &arr, int start, int end){
      int index;

      if (start < end) {
         index = partition(arr, start, end);

      #pragma omp parallel sections
      {
         #pragma omp section
         {
            quicksort(arr, start, index - 1);
         }
         #pragma omp section
         {
```

```
            quicksort(arr, index + 1, end);
          }
        }
      }
    }
}parallel;

class Serial {

    int partition(vi &arr, int start, int end){

        int pivot = arr[end];
        int i = (start - 1);

        for (int j = start; j <= end - 1; j++)
            if (arr[j] < pivot)
                swap(arr[++i], arr[j]);

        swap(arr[i + 1], arr[end]);

    return i + 1;
    }

 public:
    void quicksort(vi &arr, int start, int end){
        int index;

        if (start < end) {
            index = partition(arr, start, end);

            quicksort(arr, start, index - 1);
            quicksort(arr, index + 1, end);
        }
    }
}serial;


vd calc(int size){

    vd ans;
    vi arr(size);
    for (int i = 0; i < size; i++)
        arr[i]=rand()%size;

    vi arr2(arr);//copy the arr
```

```
    //sort the array in parallel
    double start_time = omp_get_wtime();
    parallel.quicksort(arr, 0, arr.size()-1);
    double end_time = omp_get_wtime();
    ans.push_back(end_time-start_time);

    for(auto i : arr)
        cout << i << " ";

    //sort the array in serial
    start_time = omp_get_wtime();
    serial.quicksort(arr2, 0, arr2.size() - 1);
    end_time = omp_get_wtime();
    ans.push_back(end_time-start_time);

 return ans;
}

int main(){
    cout<<"QUICK SORT IMPLEMENTATION USING OPEN MP"<<endl<<endl;
    auto _time = calc(10);

    printf("%s%32s%32s\n\n","No. of Inputs","Exec time for parallel env","Exec time for serial env");
    printf("%d%33lf%32lf\n",500,_time[0],_time[1]);
    _time = calc(1000);
    printf("%d%32lf%32lf\n",1000,_time[0],_time[1]);
    _time = calc(1200);
    printf("%d%32lf%32lf\n",1200,_time[0],_time[1]);

 return 0;
}
```

OUTPUT

```
┌──(parallels㉿gautam)-[/media/…/Desktop/projects/labs/pp]
└─$ g++ quick-sort.cpp -o qs -fopenmp

┌──(parallels㉿gautam)-[/media/…/Desktop/projects/labs/pp]
└─$ ./qs
QUICK SORT IMPLEMENTATION USING OPEN MP

No. of Inputs      Exec time for parallel env        Exec time for serial env

500                          0.000275                          0.001300
1000                         0.000670                          0.003748
1200                         0.000859                          0.002281

┌──(parallels㉿gautam)-[/media/…/Desktop/projects/labs/pp]
└─$
```