Assignment No:-02

Parallel Bubble Sort:-

```
#include<iostream>
#include<omp.h>
using namespace std;

void swap(int &a, int &b)
{
  int temp;
  temp=a;
  a=b;
  b=temp;
}
void bubble(int *a, int n)
{
  double start=omp_get_wtime();
for(int i=0;i<n;i++)
  {
    #pragma omp parallel
for(int j=i+1;j<n;j++)
    {
      if(a[j]<a[i])
      {
              swap(a[j],a[i]);
      }
    }
  }
```

```
  double end=omp_get_wtime();
  double time=end-start;
cout<<"\nTime taken => "<<time<<endl;
}
int main()
{
omp_set_num_threads(4);
  double start,end;
  int *a,n;
cout<<"\nEnter total number of elements =>";
cin>>n;
  a=new int[n];
cout<<"\nEnter elements =>";
for(int i=0;i<n;i++)
  {
            cin>>a[i];
  }
  bubble(a,n);
cout<<"\nSorted Array =>";
for(int i=0;i<n;i++)
  {
cout<<a[i]<<" ";
  }
  return 0;
}
```

OUTPUT:-

Enter total number of elements =>5

Enter elements =>11

5

22

19

13

Time taken => 0.00200009

Sorted Array =>5 11 13 19 22

Parallel Merge Sort:-

```
#include<iostream>
#include<omp.h>
using namespace std;

void merge(int *, int, int, int);
void merge_sort(int *arr, int low, int high)
{
   int mid;
   if (low < high)
   {
      mid = (low + high) / 2;
      #pragma omp parallel sections
      {
         #pragma omp section
         {
            merge_sort(arr, low, mid);
         }
         #pragma omp section
         {
            merge_sort(arr, mid + 1, high);
         }
      }
      merge(arr, low, high, mid);
   }
}

void merge(int *arr, int low, int high, int mid)
```

```
{
   int i, j, k, c[50];
   i = low;
   k = low;
   j = mid + 1;
   while (i <= mid && j <= high)
   {
      if (arr[i] < arr[j])
      {
         c[k] = arr[i];
         k++;
         i++;
      }
      else
      {
         c[k] = arr[j];
         k++;
         j++;
      }
   }
   while (i <= mid)
   {
      c[k] = arr[i];
      k++;
      i++;
   }
   while (j <= high)
   {
      c[k] = arr[j];
```

```
        k++;
        j++;
    }
    for (i = low; i < k; i++)
    {
        arr[i] = c[i];
    }
}
int main()
{
    omp_set_num_threads(4);
    int myarray[30], num;
    cout << "\nEnter number of elements to be sorted : ";
    cin >> num;
    cout << "\nEnter elements : ";
    for (int i = 0; i < num; i++)
    {
        cin >> myarray[i];
    }
    merge_sort(myarray, 0, num - 1);
    cout << "\nSorted array :" << " ";
    for (int i = 0; i < num; i++)
    {
        cout << myarray[i] << " ";
    }
}
```

OUTPUT:-

Enter number of elements to be sorted : 5

Enter elements : 99

11

23

5

6

Sorted array : 5 6 11 23 99