***Name : Kulvir Singh***

***Register Number : 19BCE2074***

***Course Code : CSE4001***

***Faculty : Dr. Vimala Devi K***

Parallel and Distributed Computing

Model Lab Fat

Question 1 :

Code :

#include<stdio.h>

#include<omp.h>

#include<stdlib.h>

int main(){

int size = 20;

int arr[size];

int i,j,first;

printf("before sorting \n");

for(i=0;i<size;i++){

arr[i] = rand();

printf("%d\n",arr[i]);

}

#pragma omp parallel shared(arr, size) private(i,j)

{

#pragma omp sections nowait

{

#pragma omp section

{

#pragma omp barrier

for(i=0;i<size-1;i++){

for(j=0;j<size-i-1;j++){

if(arr[j]>arr[j+1]){

int temp = arr[j];

arr[j] = arr[j+1];

arr[j+1] = temp;

}

}

}

}

}

}

printf("\nafter sorting \n");

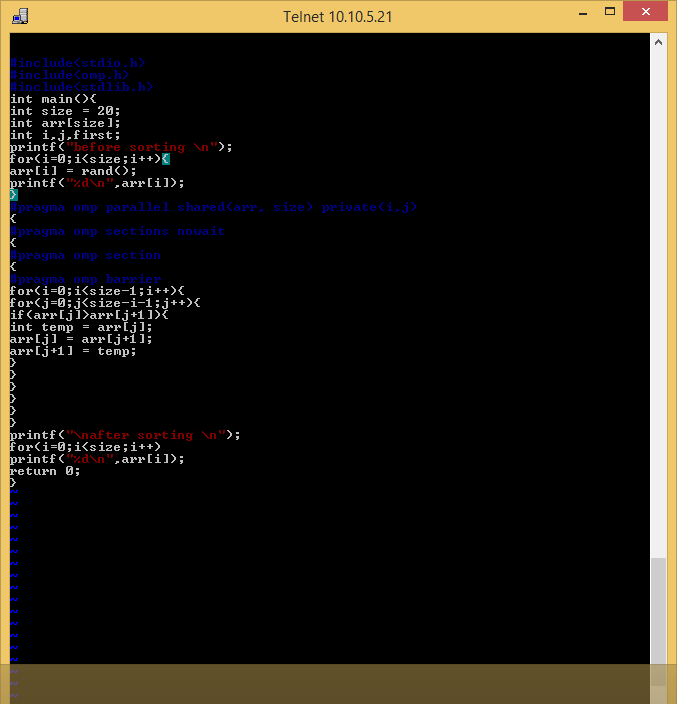
for(i=0;i<size;i++)

printf("%d\n",arr[i]);

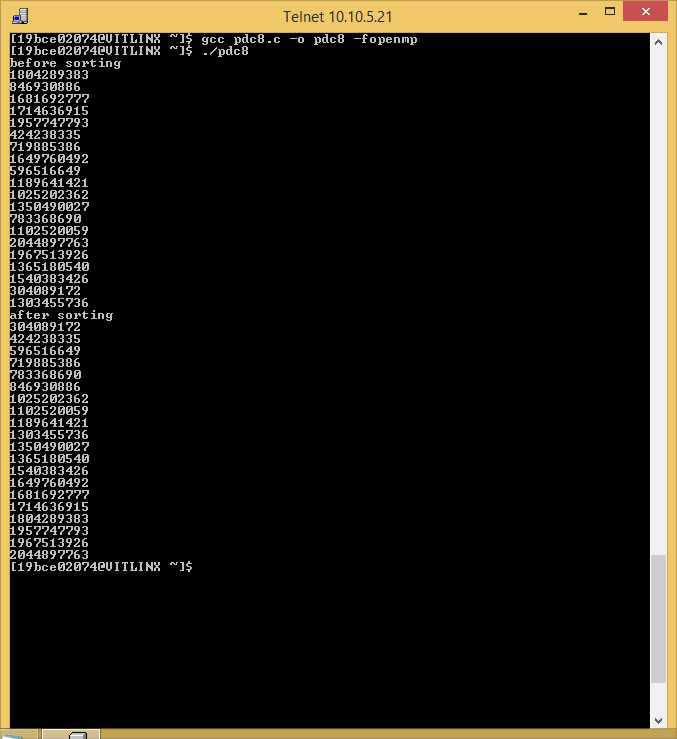
return 0;

}

**Code Screenshots :**



**Output Screenshot :**



**Question 2**

**Code :**  
#include <mpi.h>  
#include <stdio.h>

#include <stdlib.h>

#define n 10

int a[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};

int main(int argc, char \*argv[])

{

int pid, np, elements\_per\_process, n\_elements\_recieved, i;

MPI\_Status status;

MPI\_Init(&argc, &argv);

MPI\_Comm\_rank(MPI\_COMM\_WORLD, &pid);

MPI\_Comm\_size(MPI\_COMM\_WORLD, &np);

if (pid == 0)

{

int index;

elements\_per\_process = n / np;

if (np > 1)

{

for (i = 1; i < np - 1; i++)

{

index = i \* elements\_per\_process;

MPI\_Send(&elements\_per\_process, 1, MPI\_INT, i, 0, MPI\_COMM\_WORLD);

MPI\_Send(&a[index], elements\_per\_process, MPI\_INT, i, 0, MPI\_COMM\_WORLD);

}

index = i \* elements\_per\_process;

int elements\_left = n - index;

MPI\_Send(&elements\_left, 1, MPI\_INT, i, 0, MPI\_COMM\_WORLD);

MPI\_Send(&a[index], elements\_left, MPI\_INT, i, 0, MPI\_COMM\_WORLD);

}

for (i = 0; i < elements\_per\_process; i++)

sum += a[i];

int tmp;

for (i = 1; i < np; i++)

{

MPI\_Recv(&tmp, 1, MPI\_INT, MPI\_ANY\_SOURCE, 0, MPI\_COMM\_WORLD, &status);

int sender = status.MPI\_SOURCE;

sum += tmp;

}

printf("Sum of array is : %d\n", sum);

}

else

{

MPI\_Recv(&n\_elements\_recieved, 1, MPI\_INT, 0, 0, MPI\_COMM\_WORLD, &status);

MPI\_Recv(&a2, n\_elements\_recieved, MPI\_INT, 0, 0, MPI\_COMM\_WORLD, &status);

int partial\_sum = 0;

for (int i = 0; i < n\_elements\_recieved; i++)

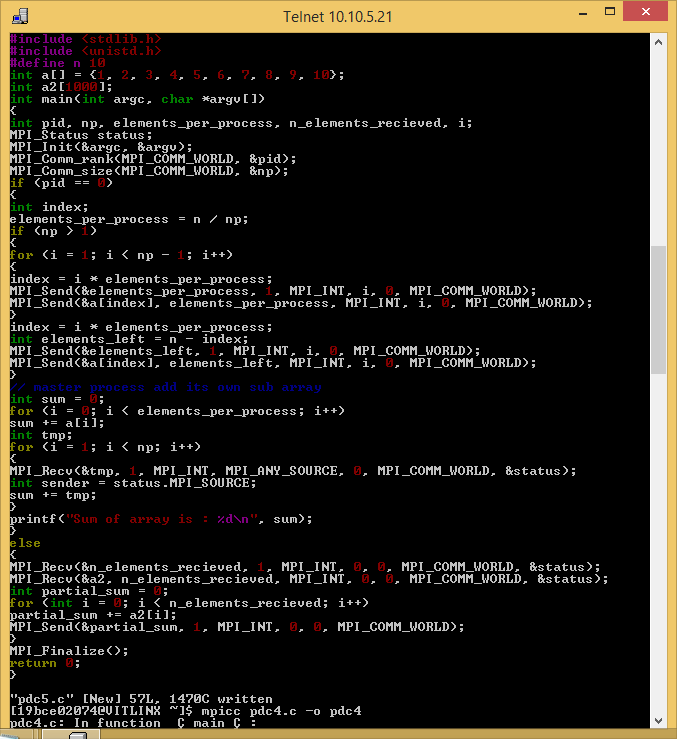
partial\_sum += a2[i];

MPI\_Send(&partial\_sum, 1, MPI\_INT, 0, 0, MPI\_COMM\_WORLD);

}

MPI\_Finalize();}

**Code Screenshot :**



**Output Screenshot :**

