**Name : Navnit Amrutharaj**

**Roll No: BIB02**

**Program:**

#include <mpi.h>

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

// size of array

#define n 10

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

// Temporary array for slave process int a2[1000];

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

{

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

// np -> no. of processes

// pid -> process id

MPI\_Status status;

// Creation of parallel processes

MPI\_Init(&argc, &argv);

// find out process ID,

// and how many processes were started

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

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

// master process if (pid == 0) {

int index, i; elements\_per\_process = n / np;

// check if more than 1 processes are run if (np > 1) {

// distributes the portion of array

// to child processes to calculate

// their partial sums 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);

printf("Server sending the elements to client: %d\n",i);

}

// last process adds remaining elements 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);

printf("Server sending the elements to client: %d\n",i);

}

// master process add its own sub array

int sum = 0;

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

sum += a[i];

printf(" Partial sum of the server : %d\n", sum);

// collects partial sums from other processes 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;

}

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

}

// slave processes else {

MPI\_Recv(&n\_elements\_recieved,

1, MPI\_INT, 0, 0,

MPI\_COMM\_WORLD,

&status);

// stores the received array segment

// in local array a2

MPI\_Recv(&a2, n\_elements\_recieved,

MPI\_INT, 0, 0,

MPI\_COMM\_WORLD,

&status);

printf("Client receiving the elements from server: %d\n",pid);

// calculates its partial sum int partial\_sum = 0;

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

partial\_sum += a2[i];

printf("Sum of array for process %d is: %d\n", pid,partial\_sum);

// sends the partial sum to the root process

MPI\_Send(&partial\_sum, 1, MPI\_INT,

0, 0, MPI\_COMM\_WORLD);

}

// cleans up all MPI state before exit of process

MPI\_Finalize();

return 0;

}

**Output:**

