**CPP LAB**

**ASSIGNMENT-5**

**NAME-**ABHISHEK TIBREWAL

**ID-**2016UCP1103

**BATCH-**A (1, 2)

**Q.Write program(s) implementing use of below MPI functions.**

**MPI\_Init**

**MPI\_Finalize**

**MPI\_Comm\_size**

**MPI\_Comm\_rank**

**MPI\_Send**

**MPI\_Recv**

**MPI\_Bcast**

**MPI\_Reduce**

**MPI\_Wtime**

**CODE1:**

#include <stdio.h>

#include <mpi.h>

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

{

MPI\_Init(&argc,&argv);

int world\_size;

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

int world\_rank;

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

float a1=MPI\_Wtime();

printf("start time for process %d is=%f\n\n",world\_rank,a1);

printf("From process %d out of %d,Hello World!\n\n", world\_rank,world\_size);

// Finalize the MPI environment.

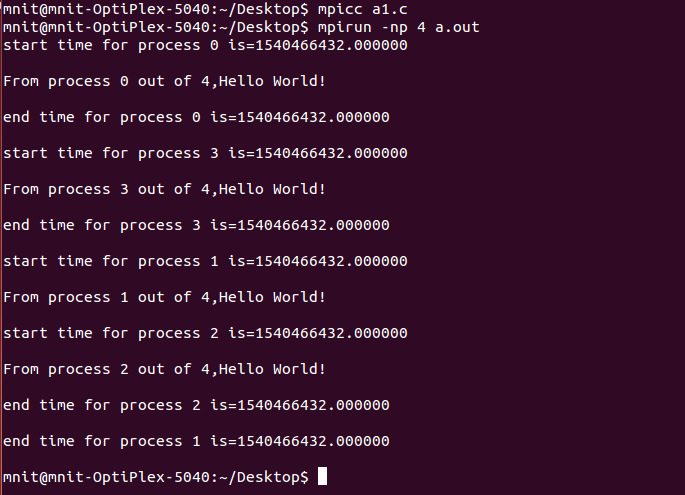
float a2=MPI\_Wtime();

printf("end time for process %d is=%f\n\n", world\_rank,a2);

MPI\_Finalize();

}

**OUTPUT:**

****

**CODE2:**

#include <stdio.h>

#include <stdlib.h>

#include <mpi.h>

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

{

int r, i, c;

float d[100],v[200];

MPI\_Status status;

MPI\_Init(&argc,&argv);

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

if(r==1)

{

for(i=0;i<100;++i)

{

d[i]=i;

}

MPI\_Send(d,100,MPI\_FLOAT,0,55,MPI\_COMM\_WORLD);

}

else

{

MPI\_Recv(v,200,MPI\_FLOAT,MPI\_ANY\_SOURCE,55,MPI\_COMM\_WORLD,&status);

printf("Process:%d Got data from processor %d \n",r,status.MPI\_SOURCE);

MPI\_Get\_count(&status,MPI\_FLOAT,&c);

printf("Process:%d Got %d elements \n",r,c);

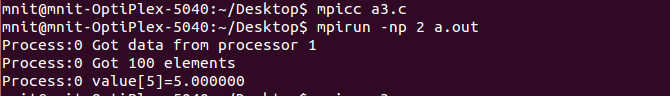
printf("Process:%d value[5]=%f \n",r,v[5]);

}

MPI\_Finalize();

}

**OUTPUT:**

****

**CODE3:**

#include <mpi.h>

#include <stdio.h>

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

{

int r;

int b;

const int rt=0;

MPI\_Init(&argc, &argv);

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

if(r == rt) {

b = 777;

}

printf("[%d]: Before Bcast, buf is %d\n", r, b);

MPI\_Bcast(&b, 1, MPI\_INT, rt, MPI\_COMM\_WORLD);

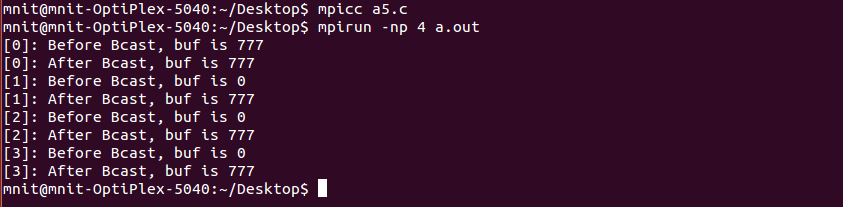
printf("[%d]: After Bcast, buf is %d\n", r, b);

MPI\_Finalize();

return 0;

}

**OUTPUT:**

****

**CODE4:**

#include<stdio.h>

#include <mpi.h>

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

{

int r;

struct {

double v;

int r;

} in, out;

int rt;

MPI\_Init(&argc, &argv);

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

in.v=r+1;

in.r=r;

rt=7;

MPI\_Reduce(&in, &out, 1, MPI\_DOUBLE\_INT, MPI\_MAXLOC, rt,MPI\_COMM\_WORLD);

if(r==rt) printf("PE:%d max=%lf at rank %d\n", r, out.v, out.r);

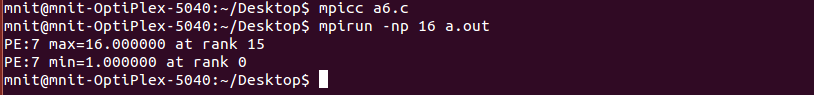
MPI\_Reduce(&in, &out, 1, MPI\_DOUBLE\_INT, MPI\_MINLOC, rt, MPI\_COMM\_WORLD);

if(r==rt) printf("PE:%d min=%lf at rank %d\n", r, out.v, out.r);

MPI\_Finalize();

}

**OUTPUT:**

****