**Lab No 2: Point to Point Communication in MPI**

1.)

#include "mpi.h"

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

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

{

int size, rank;

MPI\_Status status;

MPI\_Init(&argc, &argv);

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

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

char word[5], y[5];

int len = 5\*sizeof(char);

if (rank == 0)

{

scanf("%s", word);

MPI\_Ssend(word, len, MPI\_CHAR, 1, 101, MPI\_COMM\_WORLD);

printf("Process %d sent: %s\n", rank, word);

MPI\_Recv(word, len, MPI\_CHAR, 1, 102, MPI\_COMM\_WORLD, &status);

printf("Process %d received: %s\n", rank, word);

}

else

{

MPI\_Recv(y, len, MPI\_CHAR, 0, 101, MPI\_COMM\_WORLD, &status);

printf("Process %d received: %s\n", rank, y);

for (int i = 0; i < strlen(y); i++)

{

if (y[i] >= 'A' && y[i] <= 'Z')

y[i] += 32;

else if (y[i] >= 'a' && y[i] <= 'z')

y[i] -= 32;

}

sleep(1);

MPI\_Ssend(y, len, MPI\_CHAR, 0, 102, MPI\_COMM\_WORLD);

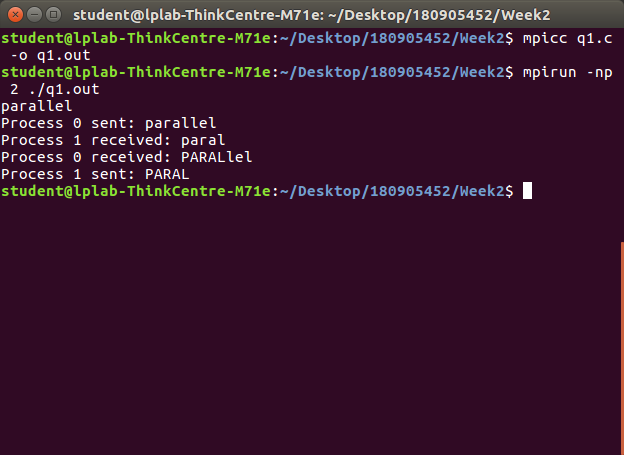
printf("Process %d sent: %s\n", rank, y);

}

MPI\_Finalize();

}

Output:



2.)

#include <mpi.h>

#include <stdio.h>

#include <stdlib.h>

#define SIZE sizeof(int)

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

{

int size, rank;

MPI\_Status status;

MPI\_Init(&argc, &argv);

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

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

int \*number = (int \*)malloc(SIZE);

int i;

if (rank == 0)

{

\*number = rand() % 10 + 1;

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

{

printf("%d. Sent to %d: %d\n", rank, i, \*number);

// Send to the process with ID = i

MPI\_Send(number, SIZE, MPI\_INT, i, 100 + i, MPI\_COMM\_WORLD);

}

}

else

{

// Revc from the process with ID = 0

MPI\_Recv(number, SIZE, MPI\_INT, 0, 100 + rank, MPI\_COMM\_WORLD, &status);

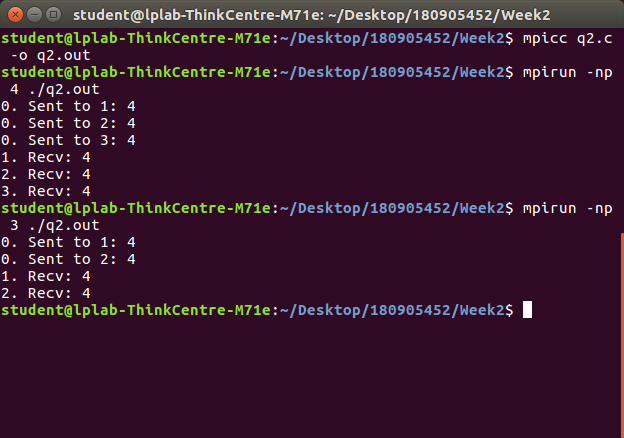
printf("%d. Recv: %d\n", rank, \*number);

}

MPI\_Finalize();

}

Output:



3.)

#include "mpi.h"

#include <stdio.h>

#include <stdlib.h>

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

int rank, size;

MPI\_Status status;

MPI\_Init(&argc, &argv);

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

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

if (rank == 0)

{

int arr[5];

for(int i=0;i<5;i++)

scanf("%d", &arr[i]);

for (int i = 1; i < size; i++)

{

MPI\_Ssend(arr + i, sizeof(int), MPI\_INT, i, 100 + i, MPI\_COMM\_WORLD);

printf("Process %d sent %d to Process %d.\n", rank, arr[i], i);

}

}

else

{

int num;

MPI\_Recv(&num, sizeof(int), MPI\_INT, 0, 100 + rank, MPI\_COMM\_WORLD, &status);

if (rank % 2 == 0)

num = num \* num;

else

num = num \* num \* num;

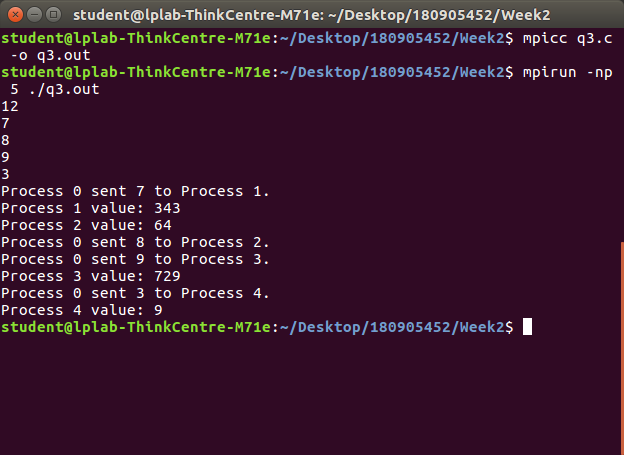
printf("Process %d value: %d\n", rank, num);

}

MPI\_Finalize();

}

Output:



4.)

#include <mpi.h>

#include <stdio.h>

#include <stdlib.h>

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

{

int size, rank;

MPI\_Status status;

MPI\_Init(&argc, &argv);

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

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

if (rank == 0)

{

int k;

fprintf(stdout, "%d. Enter a number : ", rank);

scanf\_s("%d",&k,1);

k += 1;

// Send to ID = 1

MPI\_Send(&k, 1, MPI\_INT, rank + 1, rank, MPI\_COMM\_WORLD);

fprintf(stdout, "\nNumber sent by process %d is %d.\n", rank, k);

// Receive from ID = N - 1

MPI\_Recv(&k, 1, MPI\_INT, size - 1, size - 1, MPI\_COMM\_WORLD, &status);

fprintf(stdout,"Number received by process %d is %d.\n", rank, k);

}

else

{

int k;

// Receive from ID - 1

MPI\_Recv(&k, 1, MPI\_INT, rank - 1, rank - 1, MPI\_COMM\_WORLD, &status);

fprintf(stdout,"Number received by process %d is %d.\n", rank, k);

k += 1;

// Send to (ID + 1) % size

MPI\_Send(&k, 1, MPI\_INT, (rank + 1) % size, rank, MPI\_COMM\_WORLD);

fprintf(stdout, "Number sent by process %d is %d.\n", rank, k);

}

MPI\_Finalize();

}

Output:

