2025/4/27
NO.(443814561)

Lap Project

PROBLEM2 :-

WRITE A PROGRAM (MESSAGE PASSING INTERFACE) CODE SNIPPET IN C, WHICH IMPLEMENTS A MESSAGE PASSING SCHEME IN A RING AMONG MULTIPLE PROCESSES. EACH PROCESS SENDS A MESSAGE TO ITS SUCCESSOR AND RECEIVES A MESSAGE FROM ITS PREDECESSOR IN THE RING. AFTER EXECUTION, EACH PROCESS PRINTS THE MESSAGE IT RECEIVED. ATTACH THE OUTPUT FILE (PDF) OF THE RESULT OF MESSAGE PASSING INTERFACE.

The Code:

```c
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define RING_TAG 2025

void ring_transfer(int rank, int size,
                   const char *out_msg, char *in_msg, int buf_sz)
{
    int next = (rank + 1) % size;
    int prev = (rank + size - 1) % size;
    MPI_Status status;

    MPI_Send(out_msg, (int)strlen(out_msg) + 1, MPI_CHAR, next, RING_TAG, MPI_COMM_WORLD);
    MPI_Recv(in_msg, buf_sz, MPI_CHAR, prev, RING_TAG, MPI_COMM_WORLD, &status);
}

int main(int argc, char *argv[]) {
    MPI_Init(&argc, &argv);

    int world_rank, world_size;
    MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);

    const int BUF_SIZE = 80;
    char *send_buf = malloc(BUF_SIZE);
    char *recv_buf = malloc(BUF_SIZE);
    if (!send_buf || !recv_buf) {
        fprintf(stderr, "Process %d: failed to allocate buffers\n", world_rank);
        MPI_Abort(MPI_COMM_WORLD, EXIT_FAILURE);
    }

    snprintf(send_buf, BUF_SIZE, "Greetings from node %d", world_rank);

    ring_transfer(world_rank, world_size, send_buf, recv_buf, BUF_SIZE);

    printf("Node %d/%d received: \"%s\"\n",
           world_rank, world_size, recv_buf);

    free(send_buf);
    free(recv_buf);
    MPI_Finalize();
    return EXIT_SUCCESS;}
```

The Output: