Date : 15/01/2026

PPL WEEK2 SUBMISSION

Name: Davasam Karthikeya Section: AIMLB Reg No: 230962326

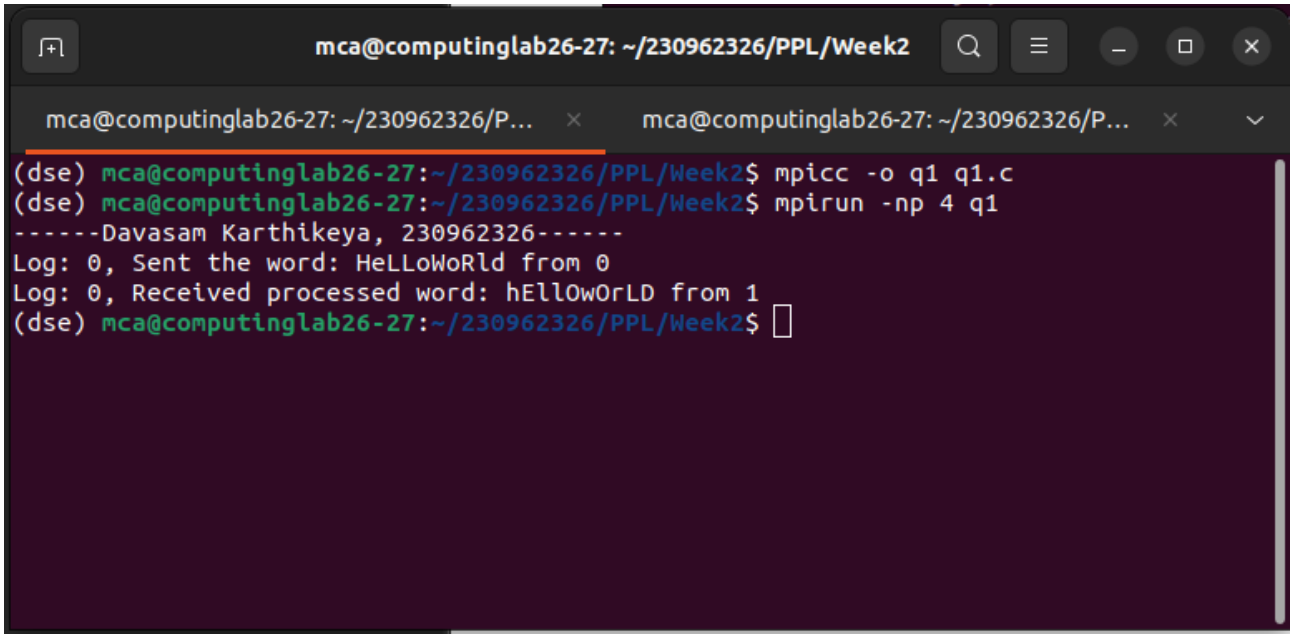
1. Write a MPI program using synchronous send. The sender process sends a word to the receiver. The second process receives the word, toggles each letter of the word and sends it back to the first process. Both processes use synchronous send operations.

Code:-

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

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

    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    MPI_Status st1;

    if(rank == 0){
        char word[] = "HeLLoWoRld";
        int word_len = strlen(word);

        printf("------Davasam Karthikeya, 230962326------\n");

        // send word count first
        MPI_Ssend(&word_len, 1, MPI_INT, 1, 0, MPI_COMM_WORLD);

        // Send Word
        MPI_Ssend(&word, word_len + 1, MPI_CHAR, 1, 0, MPI_COMM_WORLD);

        // Recv processed word
        MPI_Recv(&word, word_len + 1, MPI_CHAR, 1, 0, MPI_COMM_WORLD, &st1);
        printf("Log: %d, Received processed word: %s from 1 \n", rank, word);



    }else if(rank == 1){
        int word_len;

        // Recv Word_len
        MPI_Recv(&word_len, 1, MPI_INT, 0, 0, MPI_COMM_WORLD, &st1);

        // Recv Word
        char word[word_len + 1];
        MPI_Recv(&word, word_len + 1, MPI_CHAR, 0, 0, MPI_COMM_WORLD, &st1);

        // Toggling
        for(int i =0; i< word_len; i++)
            word[i] = (word[i] < 91) ? word[i]+32: word[i] - 32;

        // Sending it back to rank:0
```

```
        MPI_Ssend(&word, word_len + 1, MPI_CHAR, 0, 0, MPI_COMM_WORLD);
    }

    MPI_Finalize();

    return 0;
}
```

Output:-



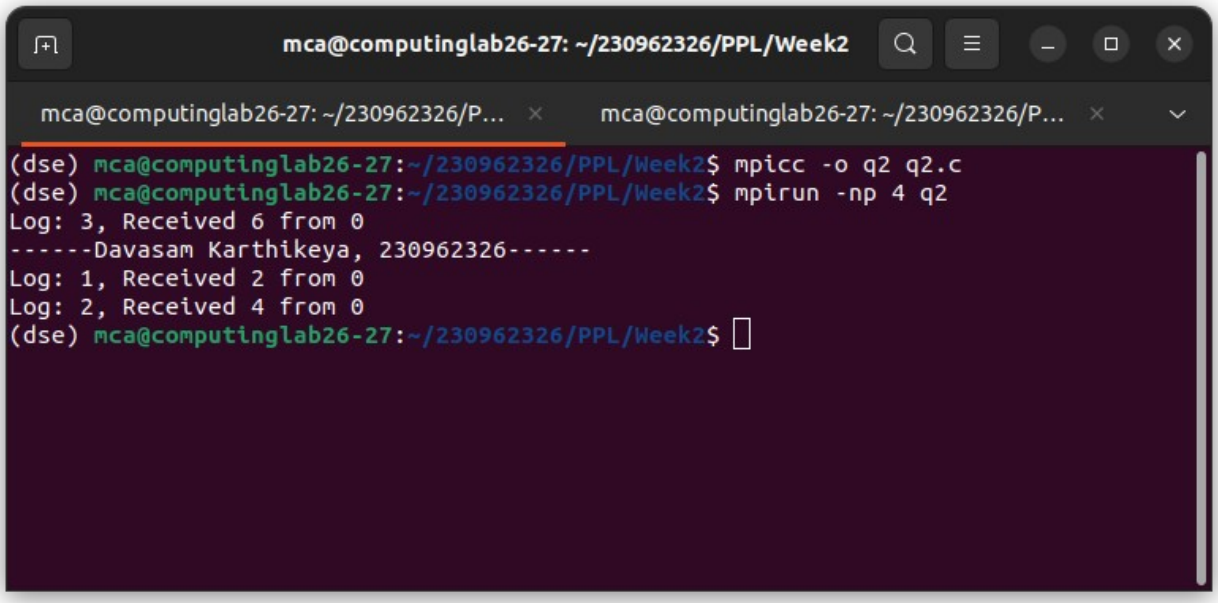
2. Write a MPI program where the master process (process 0) sends a number to each of the slaves and the slave processes receive the number and prints it. Use standard send.

Code:-

```
#include <stdio.h>
#include <mpi.h>


int main(int argc, char *argv[]){
    int size, rank;

    MPI_Init(&argc, &argv);

    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    MPI_Status st1;

    if(rank == 0){
        printf("------Davasam Karthikeya, 230962326------\n");
        for(int i = 1; i < size; i++)
            MPI_Send(&i, 1, MPI_INT, i, 0, MPI_COMM_WORLD);

    }else{
        int number;
```

```
        MPI_Recv(&number, 1, MPI_INT, 0, 0, MPI_COMM_WORLD, &st1);
        printf("Log: %d, Received %d from 0 \n", rank, number * 2);
    }

    MPI_Finalize();
}
```

Output:-



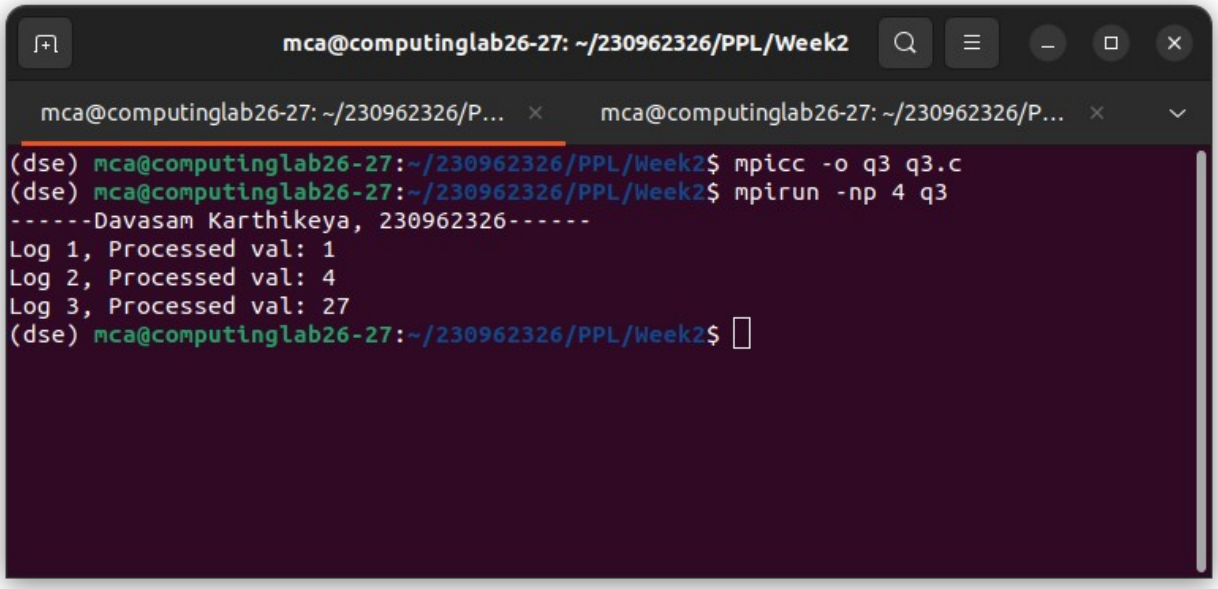
3. Write a MPI program to read N elements of the array in the root process (process 0) where N is equal to the total number of processes. The root process sends one value to each of the slaves. Let even ranked process finds square of the received element and odd ranked process finds cube of received element. Use Buffered send.

Code:-

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

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

    int size, rank;
    MPI_Init(&argc, &argv);

    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    MPI_Status st1;

    int buffer_size = MPI_BSEND_OVERHEAD + sizeof(int);
    int *buffer = (int*)malloc(buffer_size);
    MPI_Buffer_attach(buffer, buffer_size);
```

```
    if(rank == 0){
        printf("------Davasam Karthikeya, 230962326------\n");
        int data[] = {1, 2, 3, 4};
        for(int i = 1; i < size; i++)
            MPI_Bsend(&data[(i-1)%4], 1, MPI_INT, i, 0, MPI_COMM_WORLD);
    }
    else{
        int num;
        MPI_Recv(&num, 1, MPI_INT, 0, 0, MPI_COMM_WORLD, &st1);
        num = (rank%2 == 0)?num*num: num*num*num;
        printf("Log %d, Processed val: %d \n", rank, num);
    }

    MPI_Buffer_detach(&buffer, &buffer_size);
    free(buffer);

    MPI_Finalize();

    return 0;
}
```

Output:-



4. Write a MPI program to read an integer value in the root process. Root process sends this value to Processl, Process] sends this value to Process2 and so on. Last process sends the value back to root process. When sending the value each process will first increment the received value by one. Write the program using point to point communication routines.

Code:-

```
#include <stdio.h>
#include <mpi.h>

int main(int argc, char *argv[]){
    int size, rank;
```

Date : 15/01/2026

```
    MPI_Init(&argc, &argv);

    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    if(rank == 0){
        printf("------Davasam Karthikeya, 230962326------\n");
        int num;
        scanf("%d", &num);

        MPI_Ssend(&num, 1, MPI_INT, 1, 0, MPI_COMM_WORLD);
        MPI_Recv(&num, 1, MPI_INT, size-1, 0, MPI_COMM_WORLD,
MPI_STATUS_IGNORE);
        printf("Log: %d, Received back %d \n", rank, num);
    }else{
        int num;
        int dest = (rank == size-1)?0:rank+1;

        MPI_Recv(&num, 1, MPI_INT, rank-1, 0, MPI_COMM_WORLD,
MPI_STATUS_IGNORE);
        num++;
        MPI_Ssend(&num, 1, MPI_INT, dest, 0, MPI_COMM_WORLD);
    }

    MPI_Finalize();

    return 0;
}
```

Output:-