PARALLEL PROGRAMMING LAB-2 SUBMISSION

NAME : DEVADATHAN N R

ROLL NO : 04

SECTION : CSE A1

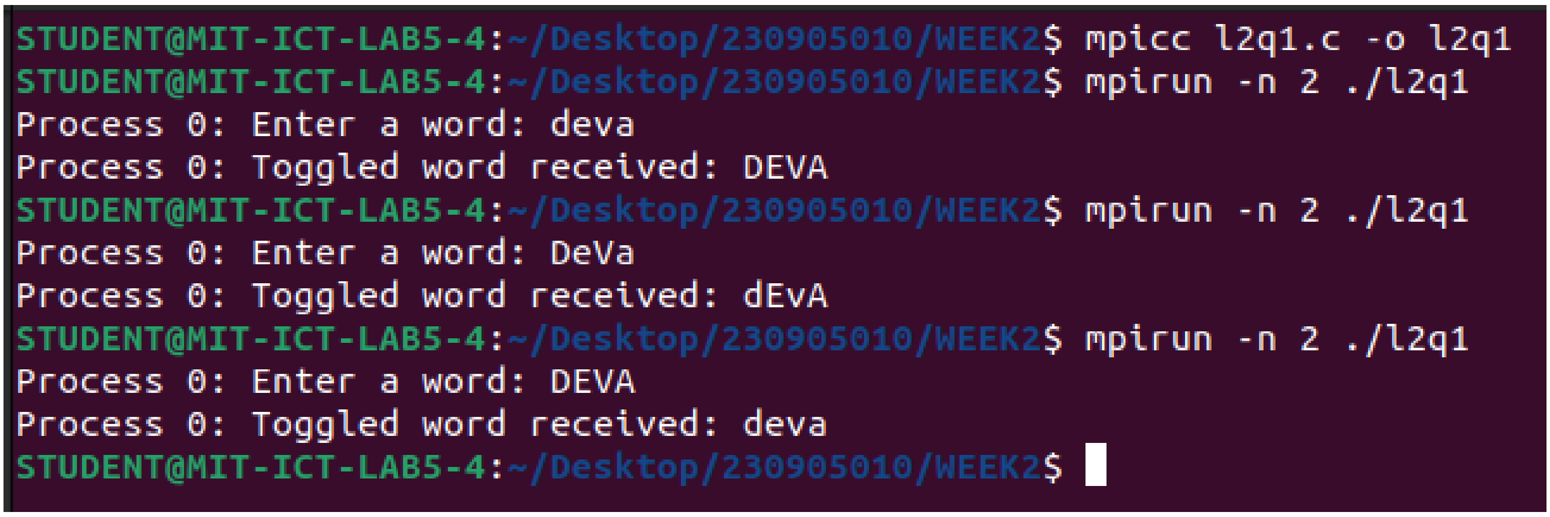
REG NO : 230905010

DATE : 16/01/26

Question 1:
MPI program using synchronous send. Sender process sends a word to 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.

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

int main(int argc, char *argv[])
{
    int rank;
    char str[100];
    MPI_Status status;
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD,&rank);

    if (rank==0)
    {
        printf("Process 0: Enter a word: ");
        scanf("%s",str);


MPI_Ssend(str,strlen(str)+1,MPI_CHAR,1,0,MPI_COMM_WORLD);

        MPI_Recv(str,100,MPI_CHAR,1,1,MPI_COMM_WORLD,&status);

        printf("Process 0: Toggled word received: %s\n",str);
    }
    else if(rank==1)
    {
        MPI_Recv(str,100,MPI_CHAR,0,0,MPI_COMM_WORLD,&status);

        for (int i=0;str[i]!='\0';i++)
        {
            if (islower(str[i]))
                str[i] = toupper(str[i]);
            else if (isupper(str[i]))
                str[i] = tolower(str[i]);
        }


MPI_Ssend(str,strlen(str)+1,MPI_CHAR,0,1,MPI_COMM_WORLD);
    }

    MPI_Finalize();
    return 0;
}
```

Output:

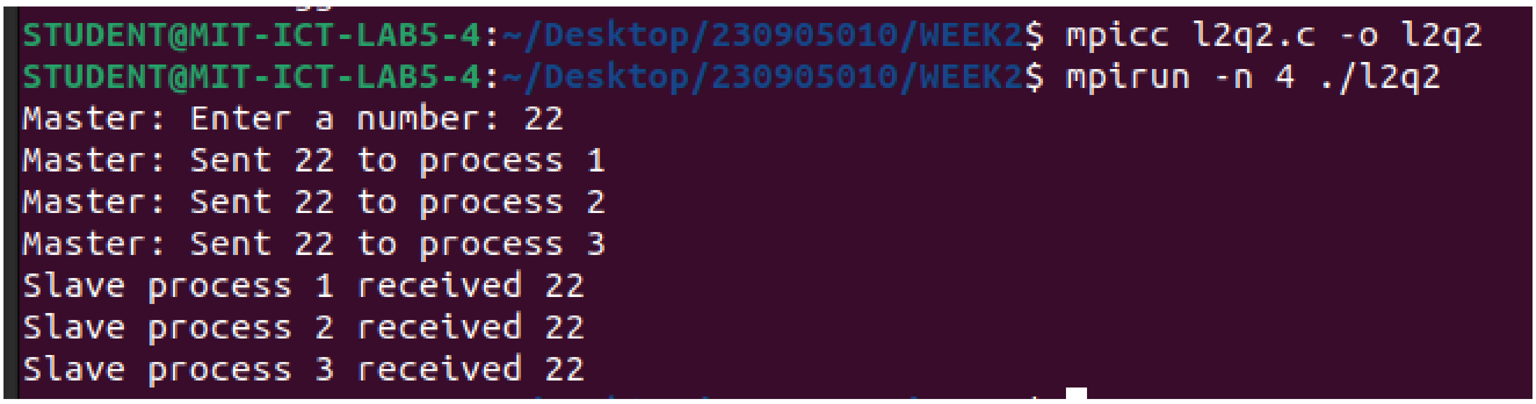


```
STUDENT@MIT-ICT-LAB5-4:~/Desktop/230905010/WEEK2$ mpicc l2q1.c -o l2q1
STUDENT@MIT-ICT-LAB5-4:~/Desktop/230905010/WEEK2$ mpirun -n 2 ./l2q1
Process 0: Enter a word: deva
Process 0: Toggled word received: DEVA
STUDENT@MIT-ICT-LAB5-4:~/Desktop/230905010/WEEK2$ mpirun -n 2 ./l2q1
Process 0: Enter a word: DeVa
Process 0: Toggled word received: dEvA
STUDENT@MIT-ICT-LAB5-4:~/Desktop/230905010/WEEK2$ mpirun -n 2 ./l2q1
Process 0: Enter a word: DEVA
Process 0: Toggled word received: deva
STUDENT@MIT-ICT-LAB5-4:~/Desktop/230905010/WEEK2$
```

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

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

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

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD,&rank);
    MPI_Comm_size(MPI_COMM_WORLD,&size);

    if (rank==0)
    {
        printf("Master: Enter a number: ");
        scanf("%d",&num);

        for (int i=1;i<size;i++)
        {
            MPI_Send(&num,1,MPI_INT,i,0,MPI_COMM_WORLD);
            printf("Master: Sent %d to process %d\n",num,i);
        }
    }
    else
    {
        MPI_Recv(&num,1,MPI_INT,0,0,MPI_COMM_WORLD,&status);
        printf("Slave process %d received %d\n",rank,num);
    }

    MPI_Finalize();
    return 0;
}
```

Output:



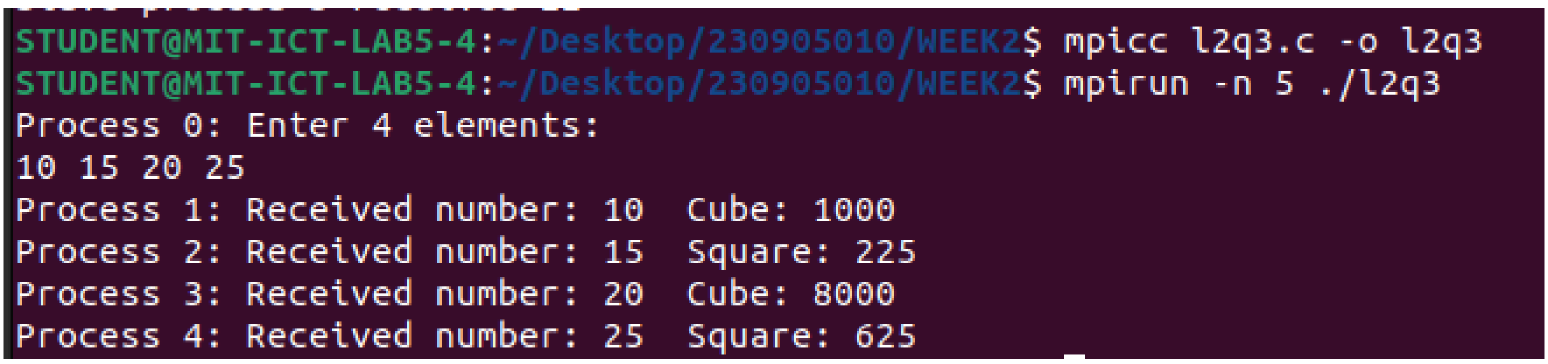
```
STUDENT@MIT-ICT-LAB5-4:~/Desktop/230905010/WEEK2$ mpicc l2q2.c -o l2q2
STUDENT@MIT-ICT-LAB5-4:~/Desktop/230905010/WEEK2$ mpirun -n 4 ./l2q2
Master: Enter a number: 22
Master: Sent 22 to process 1
Master: Sent 22 to process 2
Master: Sent 22 to process 3
Slave process 1 received 22
Slave process 2 received 22
Slave process 3 received 22
```

Question 3:
Write an MPI program to read N elements of an 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. Even ranked processes find the square of the received element and odd ranked processes find the cube of the received element. Use buffered send.

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

int main(int argc, char *argv[])
{
    int r,s,n;
    MPI_Status w;

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD,&r);
    MPI_Comm_size(MPI_COMM_WORLD,&s);

    if (r==0)
    {
        printf("Process %d: Enter %d elements:\n",r,s-1);

        int a[s];
        a[0] = 0;
    for (int i = 1; i < s; i++)
    {
        scanf("%d",&a[i]);
    }
```

```
        int bs = (sizeof(int) + MPI_BSEND_OVERHEAD) * s;
        void *b = malloc(bs);

        MPI_Buffer_attach(b, bs);

        for (int i=1;i<s;i++)
        {
            MPI_Bsend(&a[i],1,MPI_INT,i,i,MPI_COMM_WORLD);
        }

        MPI_Buffer_detach(&b,&bs);
        free(b);
    }
    else
    {
        MPI_Recv(&n,1,MPI_INT,0,r,MPI_COMM_WORLD,&w);

        if (r%2==0)
        {
            printf("Process %d: Received number: %d Square: 
%d\n",r,n,n*n);
        }
        else
        {
            printf("Process %d: Received number: %d Cube: 
%d\n",r,n,n*n*n);
        }
    }

    MPI_Finalize();
    return 0;
}
```

Output:

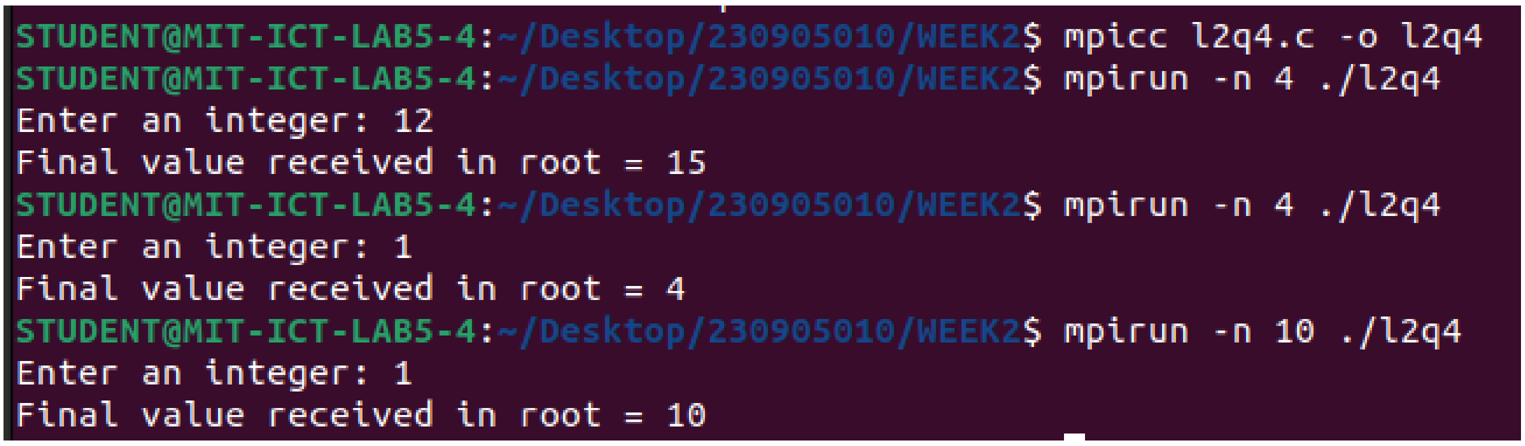


```
STUDENT@MIT-ICT-LAB5-4:~/Desktop/230905010/WEEK2$ mpicc l2q3.c -o l2q3
STUDENT@MIT-ICT-LAB5-4:~/Desktop/230905010/WEEK2$ mpirun -n 5 ./l2q3
Process 0: Enter 4 elements:
10 15 20 25
Process 1: Received number: 10  Cube: 1000
Process 2: Received number: 15  Square: 225
Process 3: Received number: 20  Cube: 8000
Process 4: Received number: 25  Square: 625
```

Question 4:
Write an MPI program to read an integer value in the root process. The root process sends this value to process 1, process 1 sends it to process 2 and so on. The last process sends the value back to the root process. When sending the value, each process increments the received value by one. Use point-to-point communication.

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

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

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD,&rank);
    MPI_Comm_size(MPI_COMM_WORLD,&size);

    if (rank==0)
    {
        printf("Enter an integer: ");
        scanf("%d",&value);

        MPI_Send(&value,1,MPI_INT,1,0,MPI_COMM_WORLD);

        MPI_Recv(&value,1,MPI_INT,size-1,0,MPI_COMM_WORLD,&status);

        printf("Final value received in root = %d\n",value);
    }
    else if (rank==size-1)
    {
        MPI_Recv(&value,1,MPI_INT,rank-1,0,MPI_COMM_WORLD,&status);
        value = value+1;
        MPI_Send(&value,1,MPI_INT,0,0,MPI_COMM_WORLD);
    }
    else
    {
        MPI_Recv(&value,1,MPI_INT,rank-1,0,MPI_COMM_WORLD,&status);
        value = value+1;
        MPI_Send(&value,1,MPI_INT,rank+1,0,MPI_COMM_WORLD);
    }

    MPI_Finalize();
    return 0;
}
```

Output:



```
STUDENT@MIT-ICT-LAB5-4:~/Desktop/230905010/WEEK2$ mpicc l2q4.c -o l2q4
STUDENT@MIT-ICT-LAB5-4:~/Desktop/230905010/WEEK2$ mpirun -n 4 ./l2q4
Enter an integer: 12
Final value received in root = 15
STUDENT@MIT-ICT-LAB5-4:~/Desktop/230905010/WEEK2$ mpirun -n 4 ./l2q4
Enter an integer: 1
Final value received in root = 4
STUDENT@MIT-ICT-LAB5-4:~/Desktop/230905010/WEEK2$ mpirun -n 10 ./l2q4
Enter an integer: 1
Final value received in root = 10
```