PPL Week-2

Rachit Takate – 230962294 – B54

I. 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 uses synchronous send operations.

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

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

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

    if(rank == 0) {
        myStr = strdup("Hello");
        len = strlen(myStr);

        MPI_Ssend(&len, 1, MPI_INT, 1, 0, MPI_COMM_WORLD);
        MPI_Ssend(myStr, len, MPI_CHAR, 1, 1, MPI_COMM_WORLD);

        MPI_Recv(myStr, len, MPI_CHAR, 1, 2, MPI_COMM_WORLD, &status);
        printf("%s\n", myStr);
    }
    else if(rank == 1) {
        MPI_Recv(&len, 1, MPI_INT, 0, 0, MPI_COMM_WORLD, &status);
        myStr = calloc(len, sizeof(char));

        MPI_Recv(myStr, len, MPI_CHAR, 0, 1, MPI_COMM_WORLD, &status);
        for(int i = 0; i < len; i++) {
            if(myStr[i] > 96)
                myStr[i] -= 32;
            else
                myStr[i] += 32;
        }

        MPI_Ssend(myStr, len, MPI_CHAR, 0, 2, MPI_COMM_WORLD);
    }

    if(rank == 0)
        printf("Rachit 230962294\n");
    MPI_Finalize();
    return 0;
}
```

Output

```
(dse) mca@computinglab26-26:~/230962294/PP Lab/Week2$ mpicc Q1.c -o Q1
(dse) mca@computinglab26-26:~/230962294/PP Lab/Week2$ mpirun -np 2 Q1
hELLO
Rachit 230962294
```

II. 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.

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

int main(int argc, char* argv[]) {
    int rank, size, val = 0;
    MPI_Init(&argc, &argv);

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

    if(rank == 0) {
        for(int i = 1; i < size; i++) {
            val += 2;
            MPI_Send(&val, 1, MPI_INT, i, 0, MPI_COMM_WORLD);
        }
    }
    else {
        MPI_Recv(&val, 1, MPI_INT, 0, 0, MPI_COMM_WORLD, &stat);
        printf("P%d val: %d\n", rank, val);
    }

    if(rank == 0)
        printf("Rachit 230962294\n");
    MPI_Finalize();
    return 0;
}
```

Output

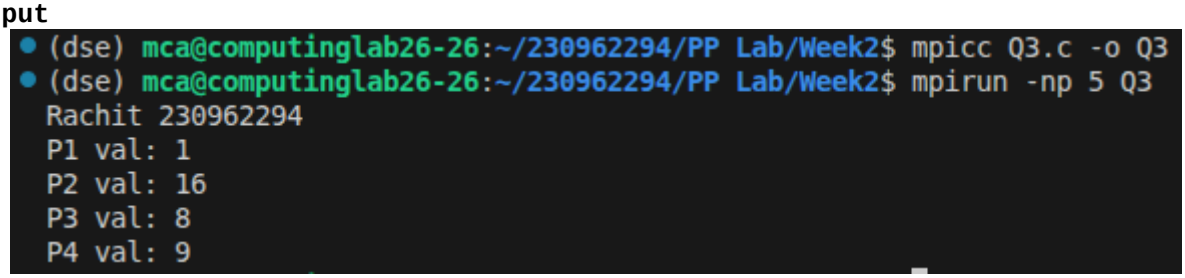
```
(dse) mca@computinglab26-26:~/230962294/PP Lab/Week2$ mpicc Q2.c -o Q2
(dse) mca@computinglab26-26:~/230962294/PP Lab/Week2$ mpirun -np 5 Q2
Rachit 230962294
P1 val: 2
P2 val: 4
P3 val: 6
P4 val: 8
```

III. 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.

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

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

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

    if(rank == 0) {
        int data[4] = {1, 4, 2, 3}, _;
        int *buff = malloc(MPI_BSEND_OVERHEAD + 4 * sizeof(int));
```

```
        MPI_Buffer_attach(buff, 4);

        for(int i = 1; i < size; i++)
            MPI_Bsend(data + (i - 1) % 4, 1, MPI_INT, i, 0, MPI_COMM_WORLD);

        MPI_Buffer_detach(buff, &_);
        free(buff);
    }
    else {
        int val;
        MPI_Recv(&val, 1, MPI_INT, 0, 0, MPI_COMM_WORLD, &stat);

        if(rank % 2)
            val *= val * val;
        else
            val *= val;

        printf("P%d val: %d\n", rank, val);
    }

    if(rank == 0)
        printf("Rachit 230962294\n");
    MPI_Finalize();
    return 0;
}
```

Output



IV. Write a MPI program to read an integer value in the root process. Root process sends this value to Process1. Process1 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.

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

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

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

    if(!rank) {
        scanf("%d", &val);
        val++;
        MPI_Ssend(&val, 1, MPI_INT, 1, 0, MPI_COMM_WORLD);

        MPI_Recv(&val, 1, MPI_INT, size - 1, 0, MPI_COMM_WORLD, &stat);
        printf("Final val: %d\n", val);
    }
    else {
```

```
        MPI_Recv(&val, 1, MPI_INT, rank - 1, 0, MPI_COMM_WORLD, &stat);
        val++;
        MPI_Ssend(&val, 1, MPI_INT, (rank + 1) % size, 0, MPI_COMM_WORLD);
    }

    if(rank == 0)
        printf("Rachit 230962294\n");
    MPI_Finalize();
    return 0;
}
```

Output