Name : Tushar Pathak
RegNumber : 230905396
Roll Number: B48

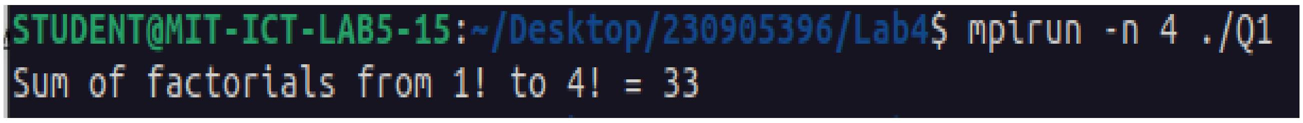
Lab 4: Collective Communications and Error Handling in MPI

Q1)Write a MPI program using N processes to find 1! +2! +.....+N!. Use scan. Also, handle different errors using error handling routines.

Code:

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

int factorial(int n) {
    int f = 1;
    for (int i = 1; i <= n; i++)
        f *= i;
    return f;
}

int main(int argc, char *argv[]) {
    int rank, size, fact, scan_sum;
    int err;
    char err_string[MPI_MAX_ERROR_STRING];
    int err_len;

    MPI_Init(&argc, &argv);

    err = MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    if (err != MPI_SUCCESS) {
        MPI_Error_string(err, err_string, &err_len);
        printf("MPI_Comm_rank error: %s\n", err_string);
        MPI_Abort(MPI_COMM_WORLD, err);
    }

    MPI_Comm_size(MPI_COMM_WORLD, &size);
    fact = factorial(rank + 1);

    err = MPI_Scan(&fact, &scan_sum, 1, MPI_INT, MPI_SUM, MPI_COMM_WORLD);
    if (err != MPI_SUCCESS) {
        MPI_Error_string(err, err_string, &err_len);
        printf("MPI_Scan error: %s\n", err_string);
        MPI_Abort(MPI_COMM_WORLD, err);
    }
    if (rank == size - 1) {
        printf("Sum of factorials from 1! to %d! = %d\n", size, scan_sum);
    }

    MPI_Finalize();
}
```

Output :



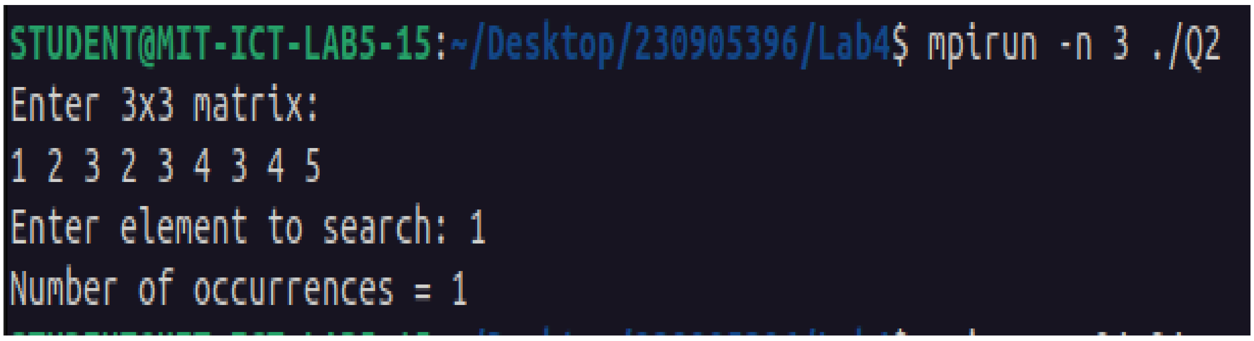
```
STUDENT@MIT-ICT-LAB5-15:~/Desktop/230905396/Lab4$ mpirun -n 4 ./Q1
Sum of factorials from 1! to 4! = 33
```

Q2)Write a MPI program to read a 3 X 3 matrix. Enter an element to be searched in the root process. Find the number of occurrences of this element in the matrix using three processes.

Code:

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

int main(int argc, char *argv[]) {
    int rank, size;
    int matrix[3][3];
    int key, count = 0, total_count;

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

    if (size != 3) {
        if (rank == 0)
            printf("Run with exactly 3 processes\n");
        MPI_Finalize();
        return 0;
    }

    if (rank == 0) {
        printf("Enter 3x3 matrix:\n");
        for (int i = 0; i < 3; i++)
            for (int j = 0; j < 3; j++)
                scanf("%d", &matrix[i][j]);

        printf("Enter element to search: ");
        scanf("%d", &key);
    }

    MPI_Bcast(matrix, 9, MPI_INT, 0, MPI_COMM_WORLD);
    MPI_Bcast(&key, 1, MPI_INT, 0, MPI_COMM_WORLD);

    for (int j = 0; j < 3; j++) {
        if (matrix[rank][j] == key)
            count++;
    }

    MPI_Reduce(&count, &total_count, 1, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);

    if (rank == 0)
        printf("Number of occurrences = %d\n", total_count);

    MPI_Finalize();
}
```

Output:



```
STUDENT@MIT-ICT-LAB5-15:~/Desktop/230905396/Lab4$ mpirun -n 3 ./Q2
Enter 3x3 matrix:
1 2 3 2 3 4 3 4 5
Enter element to search: 1
Number of occurrences = 1
```

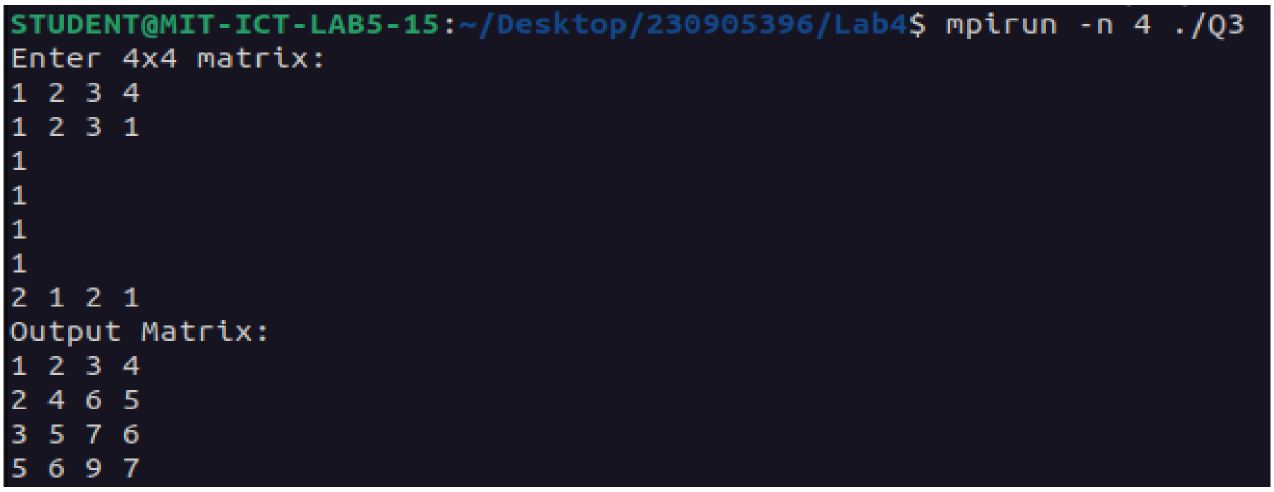
Q3)Write a MPI program to read 4 X 4 matrix and display the following output using four processes.

I/p matrix:	**1 2 3 4**	**O/p matrix:**	**1 2 3 4**
	1 2 3 1		**2 4 6 5**
	1 1 1 1		**3 5 7 6**
	2 1 2 1		**5 6 9 7**

Code:

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

int main(int argc, char *argv[]) {
    int rank, size;
    int matrix[4][4];
    int row[4], scan_row[4];
    int result[4][4];

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

    if (size != 4) {
        if (rank == 0)
            printf("Run with exactly 4 processes.\n");
        MPI_Finalize();
        return 0;
    }

    if (rank == 0) {
        printf("Enter 4x4 matrix:\n");
        for (int i = 0; i < 4; i++)
            for (int j = 0; j < 4; j++)
                scanf("%d", &matrix[i][j]);
    }

    /* Scatter one row to each process */
    MPI_Scatter(matrix, 4, MPI_INT, row, 4, MPI_INT, 0, MPI_COMM_WORLD);

    /* Cumulative column-wise sum */
    MPI_Scan(row, scan_row, 4, MPI_INT, MPI_SUM, MPI_COMM_WORLD);
    /* Gather the result matrix */
    MPI_Gather(scan_row, 4, MPI_INT, result, 4, MPI_INT, 0, MPI_COMM_WORLD);

    if (rank == 0) {
        printf("Output Matrix:\n");
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 4; j++)
                printf("%d ", result[i][j]);
            printf("\n");
        }
    }

    MPI_Finalize();
    return 0;
}
```

Output:



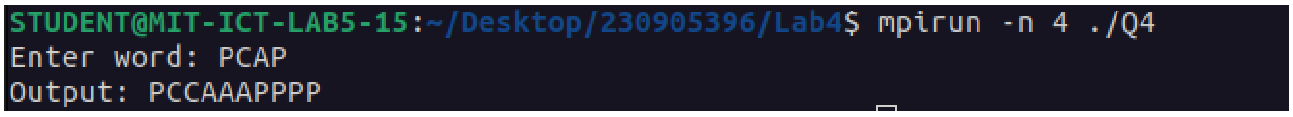
```
STUDENT@MIT-ICT-LAB5-15:~/Desktop/230905396/Lab4$ mpirun -n 4 ./Q3
Enter 4x4 matrix:
1 2 3 4
1 2 3 1
1
1
1
1
2 1 2 1
Output Matrix:
1 2 3 4
2 4 6 5
3 5 7 6
5 6 9 7
```

Q4)Write a MPI program to read a word of length N. Using N processes including the root get output word with the pattern as shown in example. Display the resultant output word in the root.
Example: Input: PCAP
Output: PCCAAAPPPP

Code:

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

int main(int argc, char *argv[]) {
    int rank, size;
    char word[100], ch;
    char local[100], result[500] = "";

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

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

    MPI_Scatter(word, 1, MPI_CHAR, &ch, 1, MPI_CHAR, 0, MPI_COMM_WORLD);

    int k = 0;
    for (int i = 0; i <= rank; i++)
        local[k++] = ch;
    local[k] = '\0';

    MPI_Gather(local, 100, MPI_CHAR, result, 100, MPI_CHAR, 0, MPI_COMM_WORLD);

    if (rank == 0) {
        printf("Output: ");
        for (int i = 0; i < size; i++)
            printf("%s", result + i * 100);
        printf("\n");
    }

    MPI_Finalize();
    return 0;
}
```

Output:



```
STUDENT@MIT-ICT-LAB5-15:~/Desktop/230905396/Lab4$ mpirun -n 4 ./Q4
Enter word: PCAP
Output: PCCAAAPPPP
```