Lab No 4: Date: 28/01/2026

Collective Communications and Error Handling

Name: Satyam Singh

Reg no: 230905256

CSE B2 - 37

Lab Exercises:

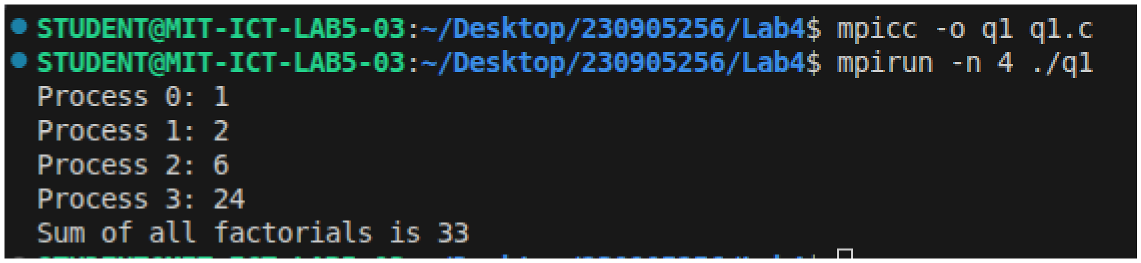
Q1) Write a MPI program using N processes to find 1! + 2! +....+N!. Use scan. Also, handle different errors using error handling routines.

Ans)

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

void ErrorHandler(int err_code) {
   if(err_code != MPI_SUCCESS) {
      char error_string[BUFSIZ];
      int length_err_string, err_class;
      MPI_Error_class(err_code, &err_class);
      MPI_Error_string(err_code, error_string, &length_err_string);
      printf("Error: %d %s\n", err_class, error_string);
   }
}
```

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

   int rank, size, factprod = 1, factsum, i, value, error_code;

   MPI_Init(&argc, &argv);

   MPI_Comm_rank(MPI_COMM_WORLD, &rank);

   MPI_Comm_size(MPI_COMM_WORLD, &size);

   MPI_Errhandler_set(MPI_COMM_WORLD, MPI_ERRORS_RETURN);


   value = rank + 1;

   error_code = MPI_Scan(&value, &factprod ,1, MPI_INT,
MPI_PROD,MPI_COMM_WORLD);

   ErrorHandler(error_code);

   printf("Process %d: %d\n", rank, factprod);


   error_code = MPI_Scan(&factprod, &factsum, 1,MPI_INT, MPI_SUM,
MPI_COMM_WORLD);

   ErrorHandler(error_code);


   if(rank == size -1) printf("Sum of all factorials is %d\n", factsum);

   MPI_Finalize();

   return 0;

}
```

Output:



```
STUDENT@MIT-ICT-LAB5-03:~/Desktop/230905256/Lab4$ mpicc -o q1 q1.c
STUDENT@MIT-ICT-LAB5-03:~/Desktop/230905256/Lab4$ mpirun -n 4 ./q1
Process 0: 1
Process 1: 2
Process 2: 6
Process 3: 24
Sum of all factorials is 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.

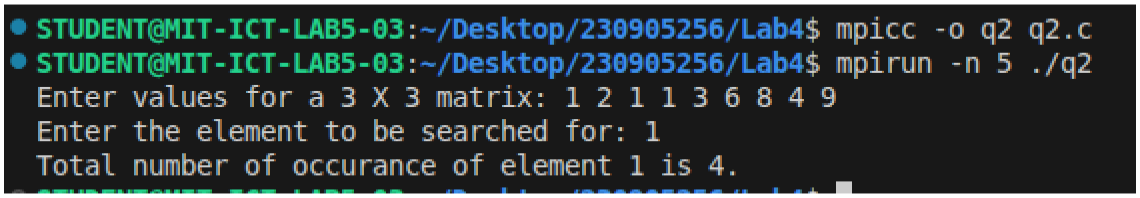
Ans)

```
#include<stdio.h>

#include "mpi.h"


void ErrorHandler(int err_code) {

  if(err_code != MPI_SUCCESS) {

    char error_string[BUFSIZ];

    int length_err_string, err_class;

    MPI_Error_class(err_code, &err_class);

    MPI_Error_string(err_code, error_string, &length_err_string);

    printf("Error: %d %s\n", err_class, error_string);

  }
```

```
}

int main(int argc, char *argv[]){
   int rank, size, error_code, mat[3][3], ele, arr[3], count = 0, ans = 0;
   MPI_Init(&argc, &argv);
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
   MPI_Comm_size(MPI_COMM_WORLD, &size);
   MPI_Errhandler_set(MPI_COMM_WORLD,MPI_ERRORS_RETURN);

   if(rank == 0){
      printf("Enter values for a 3 X 3 matrix: ");
      for(int i = 0; i < 3; i++){
         for(int j = 0; j < 3; j++){
            scanf("%d", &mat[i][j]);
         }
      }
      printf("Enter the element to be searched for: ");
      scanf("%d", &ele);
   }
   error_code = MPI_Bcast(&ele, 1, MPI_INT, 0, MPI_COMM_WORLD);
   ErrorHandler(error_code);
```

```
  error_code = MPI_Scatter(mat, 3, MPI_INT, arr, 3, MPI_INT, 0,
MPI_COMM_WORLD);
  ErrorHandler(error_code);
  for(int i = 0; i < 3; i++){
    if(arr[i] == ele) count++;
  }
  error_code = MPI_Reduce(&count, &ans, 1, MPI_INT, MPI_SUM, 0,
MPI_COMM_WORLD);
  ErrorHandler(error_code);
  if(rank == 0) printf("Total number of occurance of element %d is %d.\n",
ele, ans);
  MPI_Finalize();
  return 0;
}
```

Output:



```
STUDENT@MIT-ICT-LAB5-03:~/Desktop/230905256/Lab4$ mpicc -o q2 q2.c
STUDENT@MIT-ICT-LAB5-03:~/Desktop/230905256/Lab4$ mpirun -n 5 ./q2
Enter values for a 3 X 3 matrix: 1 2 1 1 3 6 8 4 9
Enter the element to be searched for: 1
Total number of occurance of element 1 is 4.
```

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

```
           1 2 3 1
           1 1 1 1
           2 1 2 1
```

O/p matrix : 1 2 3 4

```
           2 4 6 5
           3 5 7 6
           5 6 9 7
```

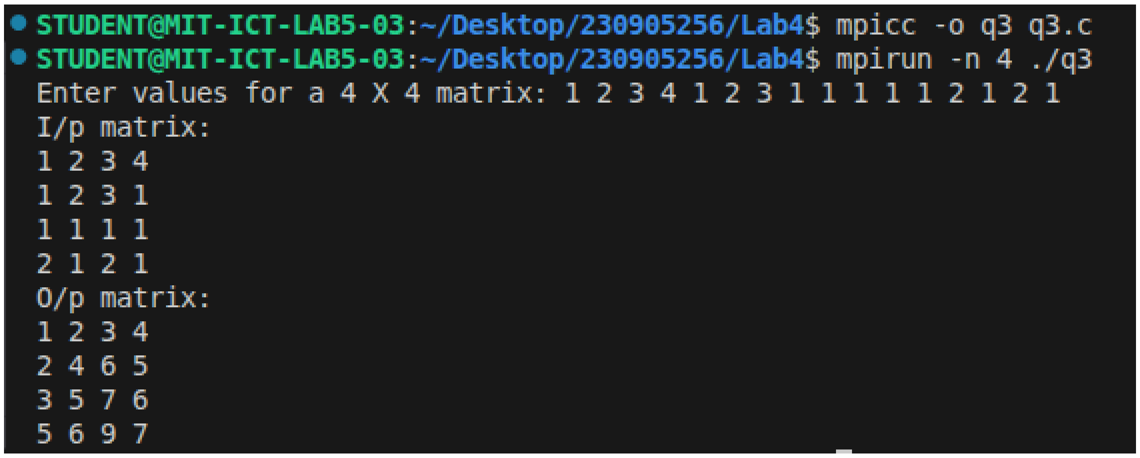
Ans)

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

void ErrorHandler(int err_code) {
   if(err_code != MPI_SUCCESS) {
      char error_string[BUFSIZ];
      int length_err_string, err_class;
      MPI_Error_class(err_code, &err_class);
```

```
        MPI_Error_string(err_code, error_string, &length_err_string);
        printf("Error: %d %s\n", err_class, error_string);
    }
}

int main(int argc, char *argv[]){
    int rank, size, error_code, mat[4][4], ans[4][4], arr1[4], arr2[4];
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Errhandler_set(MPI_COMM_WORLD,MPI_ERRORS_RETURN);

    if(rank == 0){
        printf("Enter values for a 4 X 4 matrix: ");
        for(int i = 0; i < 4; i++){
            for(int j = 0; j < 4; j++){
                scanf("%d", &mat[i][j]);
            }
        }
        printf("I/p matrix:\n");
         for(int i = 0; i < 4; i++){
```

```
        for(int j = 0; j < 4; j++){
           printf("%d ", mat[i][j]);
        }
        printf("\n");
     }
  }
  error_code = MPI_Scatter(mat, 4, MPI_INT, arr1, 4, MPI_INT,0,
MPI_COMM_WORLD);
  ErrorHandler(error_code);
  error_code = MPI_Scan(arr1, arr2, 4, MPI_INT, MPI_SUM,
MPI_COMM_WORLD);
  ErrorHandler(error_code);
  error_code = MPI_Gather(arr2, 4, MPI_INT, ans,4, MPI_INT, 0,
MPI_COMM_WORLD);
  ErrorHandler(error_code);
  if(rank == 0){
     printf("O/p matrix:\n");
      for(int i = 0; i < 4; i++){
        for(int j = 0; j < 4; j++){
           printf("%d ", ans[i][j]);
        }
        printf("\n");
     }
  }
  MPI_Finalize();
  return 0;
}
```

Output:



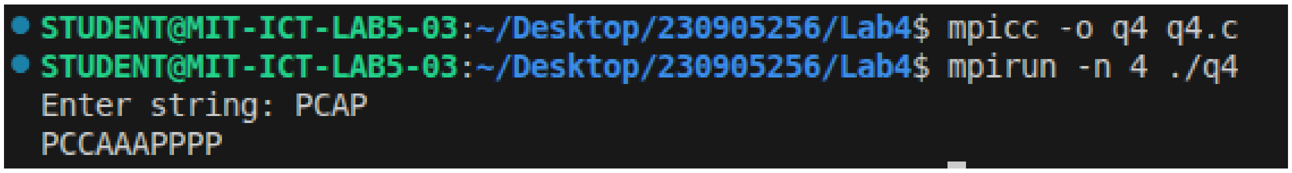
```
STUDENT@MIT-ICT-LAB5-03:~/Desktop/230905256/Lab4$ mpicc -o q3 q3.c
STUDENT@MIT-ICT-LAB5-03:~/Desktop/230905256/Lab4$ mpirun -n 4 ./q3
Enter values for a 4 X 4 matrix: 1 2 3 4 1 2 3 1 1 1 1 1 2 1 2 1
I/p matrix:
1 2 3 4
1 2 3 1
1 1 1 1
2 1 2 1
O/p matrix:
1 2 3 4
2 4 6 5
3 5 7 6
5 6 9 7
```

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

Ans)

```
#include<stdio.h>

#include "mpi.h"


void ErrorHandler(int err_code) {

   if(err_code != MPI_SUCCESS) {

      char error_string[BUFSIZ];

      int length_err_string, err_class;

      MPI_Error_class(err_code, &err_class);

      MPI_Error_string(err_code, error_string, &length_err_string);

      printf("Error: %d %s\n", err_class, error_string);

   }

}


int main(int argc, char *argv[]){
  int rank, size, error_code, num1, num2, arr[4], i=0, j=0;

  char str[4];

  MPI_Init(&argc, &argv);

  MPI_Comm_rank(MPI_COMM_WORLD, &rank);

  MPI_Comm_size(MPI_COMM_WORLD, &size);

  MPI_Errhandler_set(MPI_COMM_WORLD,MPI_ERRORS_RETURN);


  if(rank == 0){

    printf("Enter string: ");

    scanf("%s",str);

  }

  num1 = rank + 1;

  error_code = MPI_Scan(&num1, &num2, 1, MPI_INT, MPI_SUM,
MPI_COMM_WORLD);

  if(rank == 0) ErrorHandler(error_code);

  error_code = MPI_Gather(&num2, 1, MPI_INT, arr, 1, MPI_INT, 0,
MPI_COMM_WORLD);

  if(rank == 0) ErrorHandler(error_code);

  if(rank == 0){

    int totalSize = size * (size + 1) / 2;

    for(int i = 0; i <= totalSize; i++){
            if(i == arr[j]) j++;

            printf("%c", str[j]);

        }

        printf("\n");

    }

    MPI_Finalize();

    return 0;

}
```

Output:



```
STUDENT@MIT-ICT-LAB5-03:~/Desktop/230905256/Lab4$ mpicc -o q4 q4.c
STUDENT@MIT-ICT-LAB5-03:~/Desktop/230905256/Lab4$ mpirun -n 4 ./q4
Enter string: PCAP
PCCAAAPPPP
```