Lab-4 Collective Communications and Error Handling in MPI

```
// Title  : MPI Program to compute the sum of fact of ranks
// Author : Aditya Sinha
// Date   : 30/01/2026

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

int factorial(int num){
    if(num==0 || num==1){
        return 1;

    }
    return num*factorial(num-1);
}

void ErrorHandler(int error_code);

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

    int rank,size, error_code;

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

    int fact=factorial(rank+1);

    int sum;

    MPI_Errhandler_set(MPI_COMM_WORLD, MPI_ERRORS_RETURN);

    error_code=MPI_Scan(&fact, &sum, 1, MPI_INT, MPI_COMM_WORLD);
    ErrorHandler(error_code);

    if(rank == size - 1){
        fprintf(stdout,"1! + 2! +...+ %d!=%d \n", size, sum);
        fflush(stdout);
    }

    MPI_Finalize();
    return 0;
}

void ErrorHandler( int error_code){

    if(error_code != MPI_SUCCESS){
        char error_string[BUFSIZ];

        int length_of_error_string, error_class;
        MPI_Error_class(error_code, &error_class);
        MPI_Error_string(error_code, error_string,
&length_of_error_string);

        fprintf(stderr, "%d %s\n", error_class, error_string);
    }
}
```

Wrong Output

```
q1.c: In function ‘main’:
q1.c:30:9: warning: implicit declaration of function ‘MPI_Errhandler_set’;
did you mean ‘MPI_Errhandler_free’? [-Wimplicit-function-declaration]
   30 |         MPI_Errhandler_set(MPI_COMM_WORLD, MPI_ERRORS_RETURN);
      |         ^~~~~~~~~~~~~~~~~~
      |         MPI_Errhandler_free
In file included from q1.c:6:
/usr/lib/x86_64-linux-gnu/openmpi/include/mpi.h:402:47: warning: passing
argument 5 of ‘MPI_Scan’ from incompatible pointer type [-Wincompatible-
pointer-types]
  402 | #define OMPI_PREDEFINED_GLOBAL(type, global) ((type) ((void *) &
(global)))
      |
~^~~~~~~~~~~~~~~~~~~~~~~~~~~~
      |                                               |
      |                                               struct
ompi_communicator_t *
/usr/lib/x86_64-linux-gnu/openmpi/include/mpi.h:1140:24: note: in expansion
of macro ‘OMPI_PREDEFINED_GLOBAL’
 1140 | #define MPI_COMM_WORLD OMPI_PREDEFINED_GLOBAL( MPI_Comm,
ompi_mpi_comm_world)
      |                        ^~~~~~~~~~~~~~~~~~~~~~
q1.c:32:54: note: in expansion of macro ‘MPI_COMM_WORLD’
   32 |         error_code=MPI_Scan(&fact, &sum, 1, MPI_INT,
MPI_COMM_WORLD);
      |                                                      ^~~~~~~~~~~~~~
/usr/lib/x86_64-linux-gnu/openmpi/include/mpi.h:1766:59: note: expected
‘MPI_Op’ {aka ‘struct ompi_op_t *’} but argument is of type ‘struct
ompi_communicator_t *’
 1766 |                             MPI_Datatype datatype, MPI_Op op,
MPI_Comm comm);
      |                                                    ~~~~~~~^~
q1.c:32:20: error: too few arguments to function ‘MPI_Scan’
   32 |         error_code=MPI_Scan(&fact, &sum, 1, MPI_INT,
MPI_COMM_WORLD);
      |                    ^~~~~~~~
In file included from q1.c:6:
/usr/lib/x86_64-linux-gnu/openmpi/include/mpi.h:1765:20: note: declared
here
 1765 | OMPI_DECLSPEC  int MPI_Scan(const void *sendbuf, void *recvbuf, int
count,
```

Correct Output

```
1! + 2! +...+ 4!=33
```

```c
// Title  : MPI Program to count occurrences of a key in a matrix
// Author : Aditya Sinha
// Date   : 30/01/2026

#include <mpi.h>
#include <stdio.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);

    int a[9], row[3], key;
    if (rank == 0) {
        for (int i = 0; i < 9; i++) scanf("%d", &a[i]);
        scanf("%d", &key);
    }

    MPI_Bcast(&key, 1, MPI_INT, 0, MPI_COMM_WORLD);
    MPI_Scatter(a, 3, MPI_INT, row, 3, MPI_INT, 0, MPI_COMM_WORLD);

    int cnt = 0;
    for (int i = 0; i < 3; i++) if (row[i] == key) cnt++;

    int tot;
    MPI_Reduce(&cnt, &tot, 1, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);
    if (rank == 0) printf("%d\n", tot);

    MPI_Finalize();
    return 0;
}
```

Output

```
1 2 3
1 2 3
5 2 3
3
3
```

```
// Title  : MPI Program to compute the sum of cols till the current row
// Author : Aditya Sinha
// Date   : 30/01/2026

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

int main(int argc, char **argv) {
    int rank, size;
    MPI_Init(&argc, &argv);
    MPI_Comm_set_errhandler(MPI_COMM_WORLD, MPI_ERRORS_RETURN);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    int a[16], row[4], out[16], res[4];
    if (rank == 0) for (int i = 0; i < 16; i++) scanf("%d", &a[i]);

    MPI_Scatter(a, 4, MPI_INT, row, 4, MPI_INT, 0, MPI_COMM_WORLD);
    MPI_Scan(row, res, 4, MPI_INT, MPI_SUM, MPI_COMM_WORLD);
    MPI_Gather(res, 4, MPI_INT, out, 4, MPI_INT, 0, MPI_COMM_WORLD);

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

    MPI_Finalize();
    return 0;
}
```

```
1 2 3 4
2 3 4 5
5 7 8 2
1 1 1 1

1 2 3 4
3 5 7 9
8 12 15 11
9 13 16 12
```

```
// Title  : MPI Program to form a new word by repeating each character
based on its position
// Author : Aditya Sinha
// Date   : 30/01/2026

#include <stdio.h>
```

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

int main(int argc, char *argv[])
{
    int rank, size, i;
    char *word, *sub, *res;
    char ch;

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

    if (rank == 0) {
        word = malloc(size + 1);
        printf("Enter word (length %d): ", size);
        scanf("%s", word);
    }

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

    int count = rank + 1;
    sub = malloc(count);
    for (i = 0; i < count; i++)
        sub[i] = ch;

    if (rank == 0) {
        int total = size * (size + 1) / 2;
        res = malloc(total + 1);

        int offset = 0;

        for (i = 0; i < count; i++)
            res[offset++] = sub[i];

        for (i = 1; i < size; i++) {
            MPI_Recv(res + offset, i + 1, MPI_CHAR,
                     i, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
            offset += i + 1;
        }

        res[offset] = '\0';
        printf("Final word: %s\n", res);
    }
    else {
        MPI_Send(sub, count, MPI_CHAR, 0, 0, MPI_COMM_WORLD);
    }

    MPI_Finalize();
    return 0;
}
```

```
PCAP
Enter word (length 4): Final word: PCCAAAPPPP
```