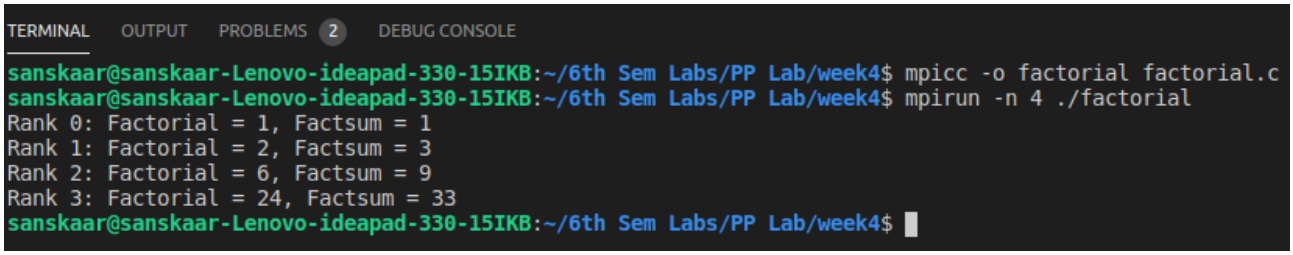
SANSKAAR PATNI
180905134 CSE-C 23
PP LAB 4
COLLECTIVE COMMUNICATION AND ERROR HANDLING

1.CODE

```
#include "mpi.h"
#include <stdio.h>
#include <string.h>
void ErrorHandler(int error_code)
{
   if (error_code != MPI_SUCCESS)
   {
      char error_string[BUFSIZ];
      int reslen, error_class;
      MPI_Error_class(error_code, &error_class);
      MPI_Error_string(error_code, error_string, &reslen);
      printf("%d %s\n", error_code, error_string);
   }
}
int main(int argc, char *argv[])
{
   int rank, size;
   MPI_Init(&argc, &argv);
   MPI_Errhandler_set(MPI_COMM_WORLD, MPI_ERRORS_RETURN);
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
   int C = 3;
   //int error_code = MPI_Comm_size(C, &size)
   int error_code = MPI_Comm_size(MPI_COMM_WORLD, &size);
   ErrorHandler(error_code);
   MPI_Status status;
   int fact = 1;
   int factsum = 0;
   for (int i = 1; i <= rank + 1; i++)
      fact *= i;
   MPI_Scan(&fact, &factsum, 1, MPI_INT, MPI_SUM, MPI_COMM_WORLD);
   printf("Rank %d: Factorial = %d, Factsum = %d\n", rank, fact, factsum);
   MPI_Finalize();
   return 0;
}
```

SCREENSHOT



2.CODE

```
#include "mpi.h"
```

```
#include <stdio.h>
#include <string.h>
void ErrorHandler(int error_code)
{
   if (error_code != MPI_SUCCESS)
   {
      char error_string[BUFSIZ];
      int reslen, error_class;
      MPI_Error_class(error_code, &error_class);
      MPI_Error_string(error_code, error_string, &reslen);
      printf("%d %s\n", error_code, error_string);
   }
   else
   {
      printf("In ErrorHandler - No errors!\n");
   }
}
int main(int argc, char *argv[])
{
   int rank, size;
   MPI_Init(&argc, &argv);
   MPI_Errhandler_set(MPI_COMM_WORLD, MPI_ERRORS_RETURN);
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
   int c = 3;
   //int error_code = MPI_Comm_size(c, &size);
   if (rank == 0)
   {
      int error_code = MPI_Comm_size(MPI_COMM_WORLD, &size);
      ErrorHandler(error_code);
   }
   MPI_Status status;
   float val = 0, pi = 0;
   val = (4.0 / (1 + ((rank + 0.5) / size) * ((rank + 0.5) / size))) * (1.0 / size);
   MPI_Reduce(&val, &pi, 1, MPI_FLOAT, MPI_SUM, 0, MPI_COMM_WORLD);
   if (rank == 0)
      printf("Rank %d: PI = %f\n", rank, pi);
   MPI_Finalize();
   return 0;
}
```

SCREENSHOT

```
sanskaar@sanskaar-Lenovo-ideapad-330-15IKB:~/6th Sem Labs/PP Lab/week4$ mpicc -o pi pi.c
sanskaar@sanskaar-Lenovo-ideapad-330-15IKB:~/6th Sem Labs/PP Lab/week4$ mpirun -n 4 ./pi
In ErrorHandler - No errors!
Rank 0: PI = 0.984615
sanskaar@sanskaar-Lenovo-ideapad-330-15IKB:~/6th Sem Labs/PP Lab/week4$
```

3.CODE

```
#include "mpi.h"
#include <stdio.h>
#include <string.h>
void ErrorHandler(int error_code)
```

```
{
   if (error_code != MPI_SUCCESS)
   {
      char error_string[BUFSIZ];
      int reslen, error_class;
      MPI_Error_class(error_code, &error_class);
      MPI_Error_string(error_code, error_string, &reslen);
      printf("%d %s\n", error_code, error_string);
   }
   else
   {
      printf("In ErrorHandler - No errors!\n");
   }
}
int main(int argc, char *argv[])
{
   int rank, size;
   MPI_Init(&argc, &argv);
   MPI_Errhandler_set(MPI_COMM_WORLD, MPI_ERRORS_RETURN);
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
   int c = 3;
   //int error_code = MPI_Comm_size(c, &size);
   int error_code = MPI_Comm_size(MPI_COMM_WORLD, &size);
   ErrorHandler(error_code);
   MPI_Status status;
   int count = 0;
   int total_count = 0;
   int matrix[3][3];
   int recvbuf[3];
   int val;
   if (rank == 0)
   {
      printf("Enter 3x3 values below:\n");
      for (int i = 0; i < 3; i++)
         for (int j = 0; j < 3; j++)
            scanf(" %d", &matrix[i][j]);
      printf("\nEnter value to search for below:\n");
      scanf(" %d", &val);
   }
   MPI_Bcast(&val, 1, MPI_INT, 0, MPI_COMM_WORLD);
   MPI_Scatter(matrix, 3, MPI_INT, recvbuf, 3, MPI_INT, 0, MPI_COMM_WORLD);
   for (int i = 0; i < 3; i++)
      if (recvbuf[i] == val)
         count++;
   MPI_Reduce(&count, &total_count, 1, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);
   if (rank == 0)
        printf("Rank %d: Total count of %d in the matrix = %d\n", rank, val, total_count);
   MPI_Finalize();
   return 0;
}
```

SCREENSHOT

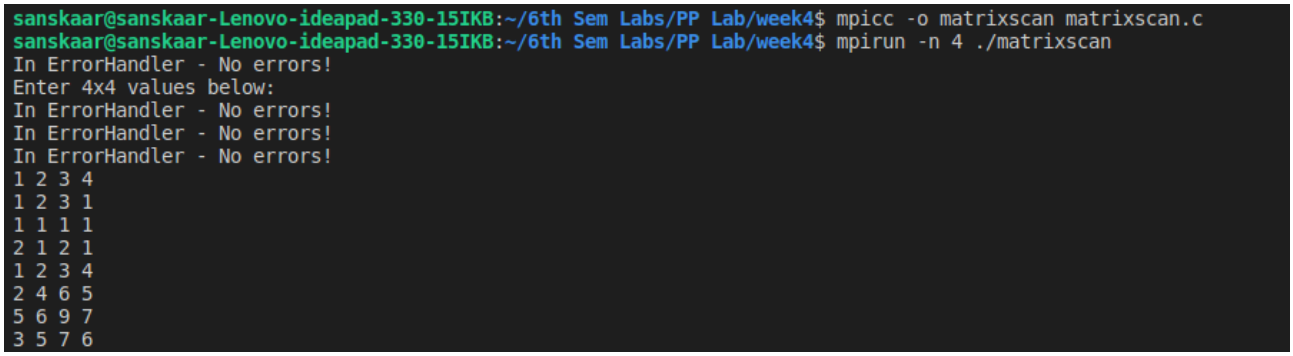
```
sanskaar@sanskaar-Lenovo-ideapad-330-15IKB:~/6th Sem Labs/PP Lab/week4$ mpirun -n 4 ./search
In ErrorHandler - No errors!
Enter 3x3 values below:
In ErrorHandler - No errors!
In ErrorHandler - No errors!
In ErrorHandler - No errors!
1 2 3
4 5 5
6 5 7

Enter value to search for below:
5
Rank 0: Total count of 5 in the matrix = 3
sanskaar@sanskaar-Lenovo-ideapad-330-15IKB:~/6th Sem Labs/PP Lab/week4$
```

4.CODE

```
#include "mpi.h"
#include <stdio.h>
#include <string.h>
void ErrorHandler(int error_code)
{
   if (error_code != MPI_SUCCESS)
   {
      char error_string[BUFSIZ];
      int reslen, error_class;
      MPI_Error_class(error_code, &error_class);
      MPI_Error_string(error_code, error_string, &reslen);
      printf("%d %s\n", error_code, error_string);
   }
   else
   {
      printf("In ErrorHandler - No errors!\n");
   }
}
int main(int argc, char *argv[])
{
   int rank, size;
   MPI_Init(&argc, &argv);
   MPI_Errhandler_set(MPI_COMM_WORLD, MPI_ERRORS_RETURN);
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
   int c = 3;
   //int error_code = MPI_Comm_size(c, &size);
   int error_code = MPI_Comm_size(MPI_COMM_WORLD, &size);
   ErrorHandler(error_code);
   MPI_Status status;
   int count = 0;
   int total_count = 0;
   int matrix[4][4];
   int recvbuf[4];
   int val;
   int sum[4];
   if (rank == 0)
   {
      printf("Enter 4x4 values below:\n");
      for (int i = 0; i < 4; i++)
```

```
            for (int j = 0; j < 4; j++)
                scanf(" %d", &matrix[i][j]);
        printf("-**-");
    }
    MPI_Scatter(matrix, 4, MPI_INT, recvbuf, 4, MPI_INT, 0,
                MPI_COMM_WORLD);
    MPI_Scan(recvbuf, sum, 4, MPI_INT, MPI_SUM, MPI_COMM_WORLD);
    for (int i = 0; i < 4; i++)
        printf("%d ", sum[i]);
    printf("\n");
    MPI_Finalize();
    return 0;
}
```

SCREENSHOT



RESPECTED SIR/MA'AM,
In all programs i have commented out the
int error_code = MPI_Comm_size(c, &size); line because at the time of compiling itself it is giving me a warning and also when i run even with the MPI_ERROR_RETURN AND MPI_Errhandler_set it still terminates the program in my machine by itself. It might be due the way Ive installed mpi in my system!

That is why Ive taken error_code= =MPI_SUCCESS for all my questions.
Ive used printf("In ErrorHandler - No errors!\n"); which shows the error handler was made to run.