190905104
lab 4

1)

```
#include <mpi.h>
#include <stdio.h>
#include <string.h>
#define BUFSIZE 100
void Error_Handler(int error_code);
int main(int argc, char *argv[])
{
   int size, rank;
   int sum = 0;
   int fact = 1;
   MPI_Init(&argc, &argv);
   MPI_Errhandler_set(MPI_COMM_WORLD, MPI_ERRORS_RETURN);
   int error_code;
   error_code = MPI_Comm_size(MPI_COMM_WORLD, &size);
   error_code = MPI_Comm_rank(MPI_COMM_WORLD, &rank);
   Error_Handler(error_code);
   int sendval = rank + 1;
   MPI_Scan(&sendval, &fact, 1, MPI_INT, MPI_PROD, MPI_COMM_WORLD);
   fprintf(stdout, "process[%d]: fact is %d\n", rank, fact);
   fflush(stdout);
   MPI_Scan(&fact, &sum, 1, MPI_INT, MPI_SUM, MPI_COMM_WORLD);
   if (rank == size - 1)
      printf("Answer is: %d\n", sum);
   return 0;
}
void Error_Handler(int error_code)
{
   if (error_code != MPI_SUCCESS)
   {
      char error_string[BUFSIZE];
      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);
      printf("%d %s\n", error_class, error_string);
   }
}
```

```
student@dblab-hp-280-10:~/190905104_ParthShukla_PCAP/week4$ mpicc q1.c
student@dblab-hp-280-10:~/190905104_ParthShukla_PCAP/week4$ mpirun -n 5 ./a.out
process[0]: fact is 1
process[4]: fact is 120
process[1]: fact is 2
process[2]: fact is 6
process[3]: fact is 24
Answer is: 153
student@dblab-hp-280-10:~/190905104_ParthShukla_PCAP/week4$
```

2)

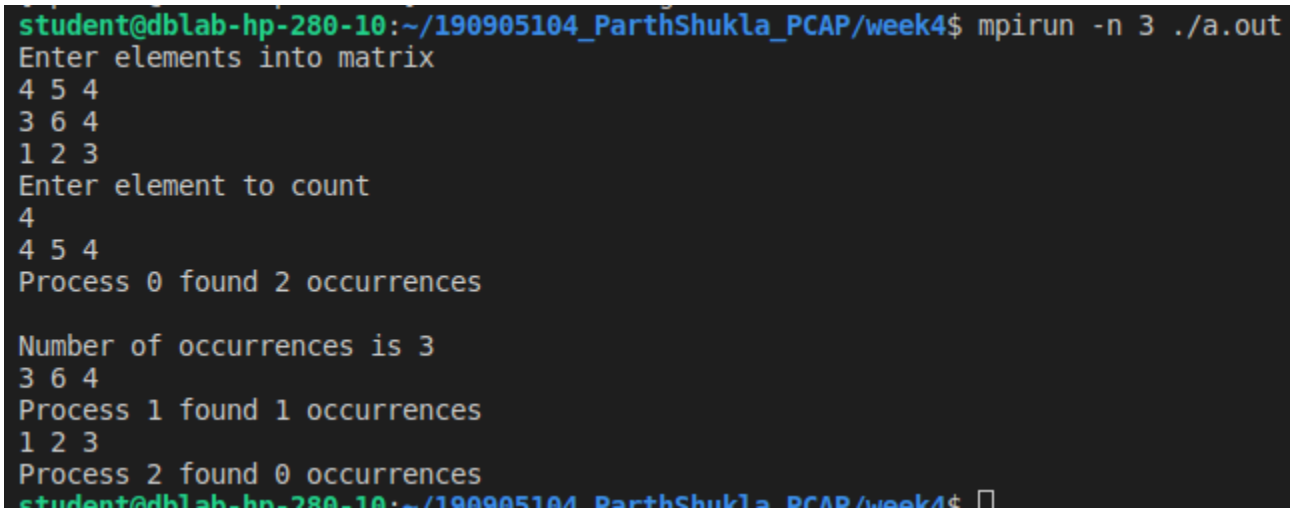
```
#include <mpi.h>
#include <stdio.h>
#include <string.h>
int main(int argc, char *argv[])
{
   int rank, size;
   int i = 0, j;
   int k = 0, fac = 1, ans[1000], sum = 0;
   int n, a[100][100], b[100];
   float x, y, area, pi1;
   MPI_Init(&argc, &argv);
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
   // Set the error handler to MPI_ERRORS_RETURN
   MPI_Errhandler_set(MPI_COMM_WORLD, MPI_ERRORS_RETURN);
   MPI_Comm_size(MPI_COMM_WORLD, &size);
   int error = MPI_Bcast(&size, 1, MPI_INT, 0, MPI_COMM_WORLD);
   if (error != MPI_SUCCESS)
   {
      char s[100];
      int len, class1;
      MPI_Error_string(error, s, &len);
      MPI_Error_class(error, &class1);
      fprintf(stderr, "Error description is %s", s);
      fflush(stderr);
      fprintf(stderr, "Error class is %d", class1);
      fflush(stderr);
   }
   x = (float)(rank + 1) / size;
   y = 4.f / (1 + x * x);
   area = (1 / (float)size) * y;
   MPI_Reduce(&area, &pi1, 1, MPI_FLOAT, MPI_SUM, 0, MPI_COMM_WORLD);
   if (rank == 0)
   {
      fprintf(stdout, "%f\n", pi1);
      fflush(stdout);
   }
   MPI_Finalize();
   return 0;
}
```

```
student@dblab-hp-280-10:~/190905104_ParthShukla_PCAP/week4$ mpicc q2.c
student@dblab-hp-280-10:~/190905104_ParthShukla_PCAP/week4$ mpirun -n 5 ./a.out
2.934926
student@dblab-hp-280-10:~/190905104_ParthShukla_PCAP/week4$ mpirun -n 10 ./a.out
3.039926
student@dblab-hp-280-10:~/190905104_ParthShukla_PCAP/week4$ mpirun -n 50 ./a.out
3.121526
student@dblab-hp-280-10:~/190905104_ParthShukla_PCAP/week4$ mpirun -n 100 ./a.out
3.131576
student@dblab-hp-280-10:~/190905104_ParthShukla_PCAP/week4$ mpirun -n 200 ./a.out
3.136589
student@dblab-hp-280-10:~/190905104_ParthShukla_PCAP/week4$ mpirun -n 300 ./a.out
3.138258
```

3)

```
#include <stdio.h>
#include <mpi.h>
void ErrorHandler(int error_code)
{
   char error_string[MPI_MAX_ERROR_STRING];
   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);
   if (error_code != 0)
      printf("error class %d \n error string %s\n", error_class, error_string);
}
void main(int a, char *b[])
{
   int rank, ele, size;
   int a1[3][3];
   int b1[3];
   int count = 0;
   int tc = 0;
   int error_code;
   MPI_Init(&a, &b);
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
   MPI_Comm_size(MPI_COMM_WORLD, &size);
   MPI_Errhandler_set(MPI_COMM_WORLD, MPI_ERRORS_RETURN);
   error_code = MPI_Comm_size(MPI_COMM_WORLD, &size);
   ErrorHandler(error_code);
   if (rank == 0)
   {
      printf("Enter elements into matrix\n");
      for (int i = 0; i < 3; i++)
      {
         for (int j = 0; j < 3; j++)
         {
            scanf("%d", &a1[i][j]);
         }
      }
      printf("Enter element to count \n");
      scanf("%d", &ele);
   }
   MPI_Bcast(&ele, 1, MPI_INT, 0, MPI_COMM_WORLD);
   MPI_Scatter(a1, 3, MPI_INT, b1, 3, MPI_INT, 0, MPI_COMM_WORLD);
   for (int i = 0; i < 3; i++)
   {
      printf("%d ", b1[i]);
      if (b1[i] == ele)
      {
         count++;
      }
   }
   printf("\nProcess %d found %d occurrences\n", rank, count);
```

```
    MPI_Reduce(&count, &tc, 1, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);
    if (rank == 0)
    {
        printf("\nNumber of occurrences is %d\n", tc);
    }
    MPI_Finalize();
}
```



4)

```
#include <mpi.h>
#include <stdio.h>
#include <string.h>
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);
        printf("%d %s\n", error_class, error_string);
    }
}
int main(int argc, char *argv[])
{
    int rank, size, error_code;
    int i = 0, j;
    int k = 0, fac = 1, ans[1000], sum = 0;
    int n, a[100][100], b[100];
    MPI_Init(&argc, &argv);
    error_code = MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    error_code = MPI_Comm_size(MPI_COMM_WORLD, &size);
    if (rank == 0)
    {
```

```
        printf("Enter the elements of i/p matrix \n");
        for (i = 0; i < 4; i++)
        {
            for (j = 0; j < 4; j++)
            {
                scanf("%d", &a[i][j]);
            }
        }
        printf("\n");
    }
    error_code = MPI_Scatter(a, 100, MPI_INT, b, 100, MPI_INT, 0, MPI_COMM_WORLD);
    error_code = MPI_Scan(b, ans, 4, MPI_INT, MPI_SUM, MPI_COMM_WORLD);
    ErrorHandler(error_code);
    for (i = 0; i < 4; i++)
    {
        printf("%d ", ans[i]);
    }
    printf("\n");
    MPI_Finalize();
    return 0;
}
```

```
student@dblab-hp-280-10:~/190905104_ParthShukla_PCAP/week4$ mpirun -n 4 ./a.out
Enter the elements of i/p matrix
1 2 3 4
1 2 3 1
1 1 1 1
5 4 2 1

1 2 3 4
2 4 6 5
3 5 7 6
8 9 9 7
```