WEEK 4

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

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

int main(int argc, char *argv[])
{
   int rank, size, n;
   unsigned long long local_fact, scan_fact, local_sum, global_sum;
   int err;
   char err_str[MPI_MAX_ERROR_STRING];
   int err_len;

   err = MPI_Init(&argc, &argv);
   if (err != MPI_SUCCESS)
   {
      MPI_Error_string(err, err_str, &err_len);
      printf("%s\n", err_str);
      MPI_Abort(MPI_COMM_WORLD, err);
   }

   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
   MPI_Comm_size(MPI_COMM_WORLD, &size);

   if (rank == 0)
   {
      if (argc != 2)
      {
         printf("Usage: mpirun -np N ./a.out <N>\n");
         MPI_Abort(MPI_COMM_WORLD, 1);
      }
      n = atoi(argv[1]);
      if (n <= 0 || n != size)
      {
         printf("N must be positive and equal to number of processes\n");
         MPI_Abort(MPI_COMM_WORLD, 1);
      }
   }

   MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);

   local_fact = rank + 1;

   MPI_Scan(&local_fact, &scan_fact, 1, MPI_UNSIGNED_LONG_LONG, MPI_PROD,
MPI_COMM_WORLD);

   local_sum = scan_fact;
```

```
    MPI_Reduce(&local_sum, &global_sum, 1, MPI_UNSIGNED_LONG_LONG, MPI_SUM, 0,
MPI_COMM_WORLD);

    if (rank == 0)
        printf("Result = %llu\n", global_sum);

    MPI_Finalize();
    return 0;
}
```

```
STUDENT@MIT-ICT-LAB5-24:~/Desktop/230905480/week4/q1$ mpirun -np 5 ./1
Sum of factorials = 153
```

2) 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.

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

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

    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("This program requires 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", &search);
    }
```

```
    MPI_Bcast(&search, 1, MPI_INT, 0, MPI_COMM_WORLD);

    MPI_Scatter(matrix, 3, MPI_INT,
                row, 3, MPI_INT,
                0, MPI_COMM_WORLD);

    for (int i = 0; i < 3; i++)
    {
        if (row[i] == search)
            local_count++;
    }

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

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

    MPI_Finalize();
    return 0;
}
```

```
STUDENT@MIT-ICT-LAB5-24:~/Desktop/230905480/week4/q2$ mpirun -np 3 ./2
Enter 3x3 matrix:
1 2 3
4 5 6
1 2 2
Enter element to search: 2
Number of occurrences of 2 = 3
```

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

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

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

  MPI_Scatter(matrix, 4, MPI_INT,
              row, 4, MPI_INT,
              0, MPI_COMM_WORLD);

  MPI_Scan(row, scan_row, 4, MPI_INT,
           MPI_SUM, MPI_COMM_WORLD);

  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;
}
```

```
STUDENT@MIT-ICT-LAB5-24:~/Desktop/230905480/week4/q3$ mpirun -np 4 ./3
Enter 4x4 matrix:
1 2 3 4
5 6 7 8
1 2 3 4
1 1 1 1
Output Matrix:
1 2 3 4
6 8 10 12
7 10 13 16
8 11 14 17
```

4) 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.

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

int main(int argc, char *argv[])
{
   int rank, size, len;
   char word[100];

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

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

   MPI_Bcast(&len, 1, MPI_INT, 0, MPI_COMM_WORLD);

   if (size != len)
   {
      if (rank == 0)
         printf("Number of processes must be equal to word length\n");
      MPI_Finalize();
      return 0;
   }

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

   char local[100];
   int count = rank + 1;

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

   int recvcounts[100], displs[100];
   int total = 0;

   if (rank == 0)
   {
      for (int i = 0; i < size; i++)
```

```
        {
            recvcounts[i] = i + 1;
            displs[i] = total;
            total += recvcounts[i];
        }
    }

    char result[500];

    MPI_Gatherv(local, count, MPI_CHAR,
                result, recvcounts, displs,
                MPI_CHAR, 0, MPI_COMM_WORLD);

    if (rank == 0)
    {
        result[total] = '\0';
        printf("Output: %s\n", result);
    }

    MPI_Finalize();
    return 0;
}
```

```
STUDENT@MIT-ICT-LAB5-24:~/Desktop/230905480/week4/q4$ mpirun -np 3 ./4
Enter the word: ABC
Output: ABBCCC
```