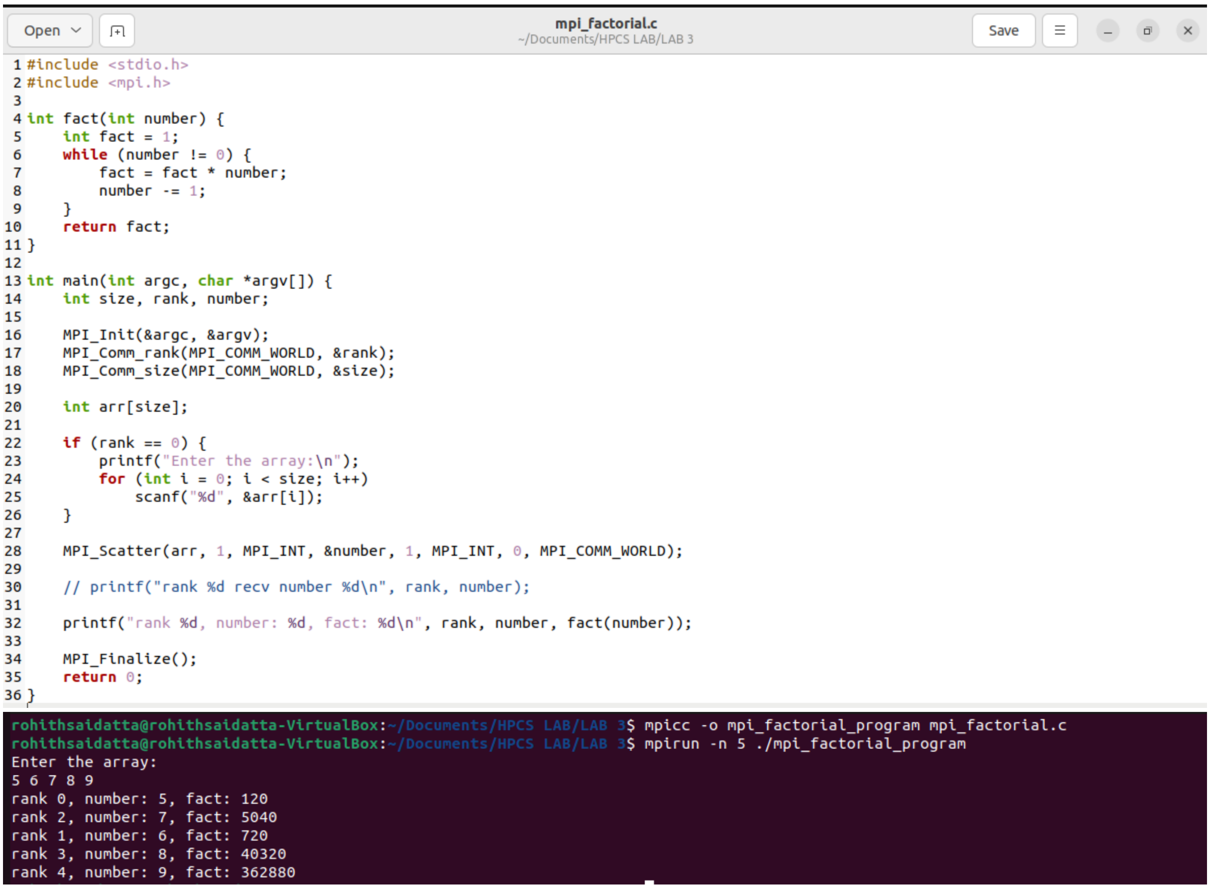
Lab No. 3 Week 3

MPI PROGRAMMING

USE OF MPI_Beast, MPI Scatter and MPI_Gather

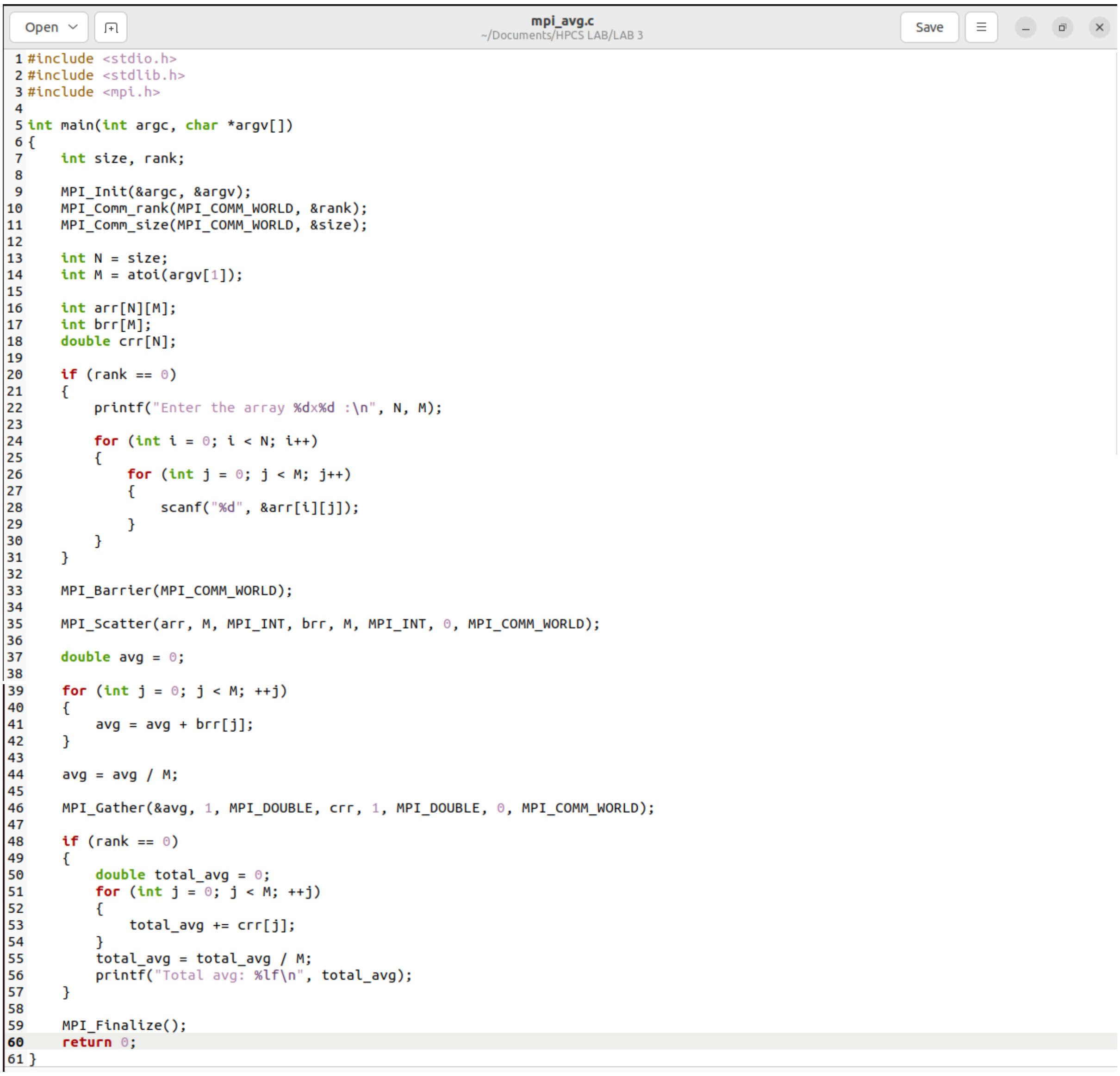
1) Write a MPI program to read N values in the root process. Root process sends one value to each process. Every process receives it prints the factorial of that number. Use N number of processes.



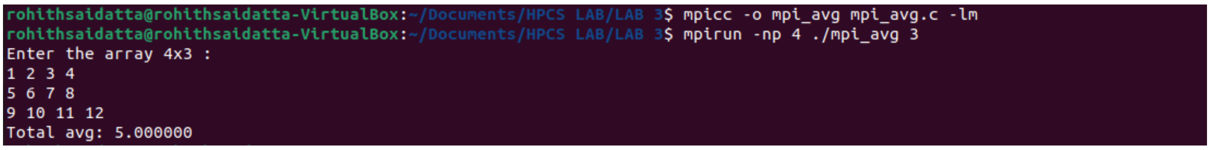
```c
#include <stdio.h>
#include <mpi.h>

int fact(int number) {
    int fact = 1;
    while (number != 0) {
        fact = fact * number;
        number -= 1;
    }
    return fact;
}

int main(int argc, char *argv[]) {
    int size, rank, number;

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

    int arr[size];

    if (rank == 0) {
        printf("Enter the array:\n");
        for (int i = 0; i < size; i++)
            scanf("%d", &arr[i]);
    }

    MPI_Scatter(arr, 1, MPI_INT, &number, 1, MPI_INT, 0, MPI_COMM_WORLD);

    // printf("rank %d recv number %d\n", rank, number);

    printf("rank %d, number: %d, fact: %d\n", rank, number, fact(number));

    MPI_Finalize();
    return 0;
}
```

```
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/HPCS LAB/LAB 3$ mpicc -o mpi_factorial_program mpi_factorial.c
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/HPCS LAB/LAB 3$ mpirun -n 5 ./mpi_factorial_program
Enter the array:
5 6 7 8 9
rank 0, number: 5, fact: 120
rank 2, number: 7, fact: 5040
rank 1, number: 6, fact: 720
rank 3, number: 8, fact: 40320
rank 4, number: 9, fact: 362880
```

2) Write an MPI program to read a value M and N x M elements in the root process. The root process sends M elements to each process. Each process finds an average of M elements it receives and sends these average values to the root. Root collects all the values and finds the total average Use N number of processes.

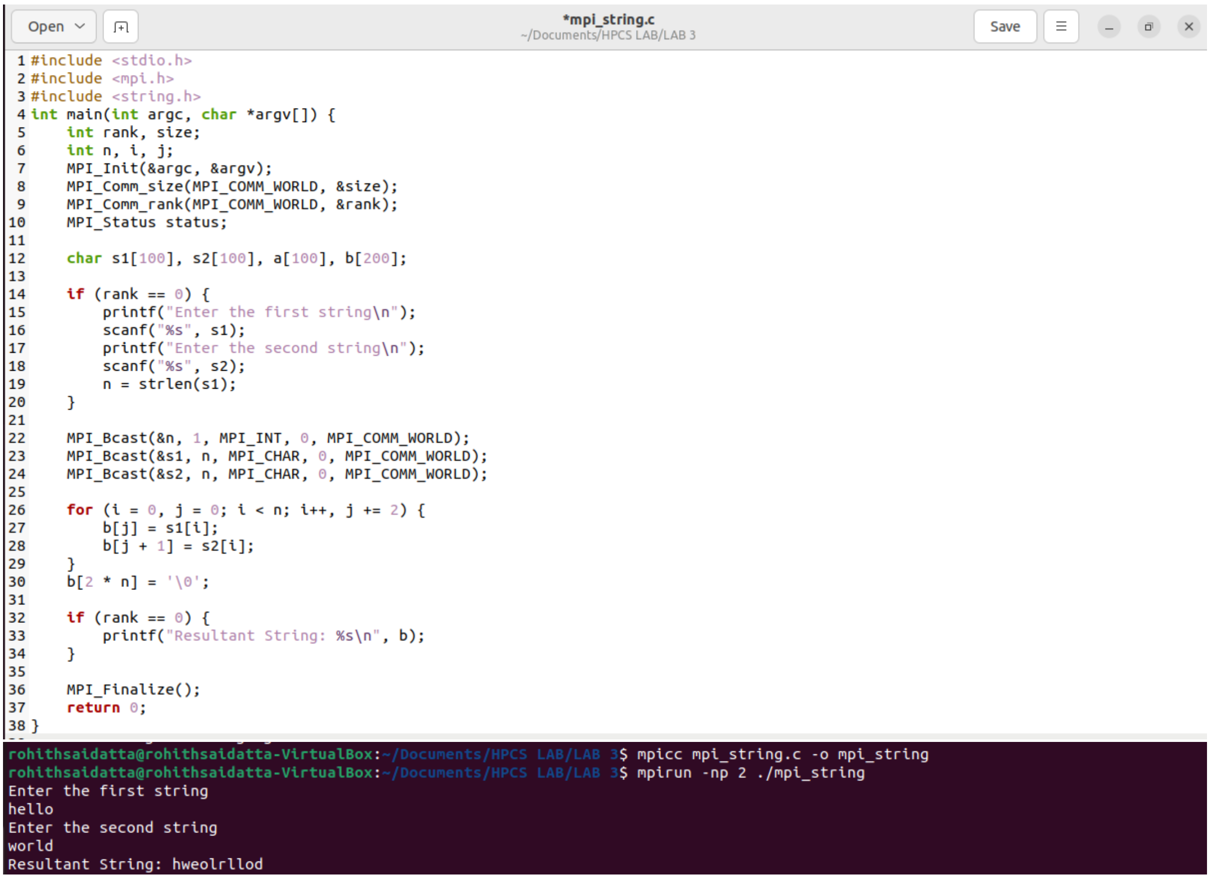


```c
#include <stdio.h>
#include <stdlib.h>
#include <mpi.h>

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

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

    int N = size;
    int M = atoi(argv[1]);

    int arr[N][M];
    int brr[M];
    double crr[N];

    if (rank == 0)
    {
        printf("Enter the array %dx%d :\n", N, M);

        for (int i = 0; i < N; i++)
        {
            for (int j = 0; j < M; j++)
            {
                scanf("%d", &arr[i][j]);
            }
        }
    }

    MPI_Barrier(MPI_COMM_WORLD);

    MPI_Scatter(arr, M, MPI_INT, brr, M, MPI_INT, 0, MPI_COMM_WORLD);

    double avg = 0;

    for (int j = 0; j < M; ++j)
    {
        avg = avg + brr[j];
    }

    avg = avg / M;

    MPI_Gather(&avg, 1, MPI_DOUBLE, crr, 1, MPI_DOUBLE, 0, MPI_COMM_WORLD);

    if (rank == 0)
    {
        double total_avg = 0;
        for (int j = 0; j < M; ++j)
        {
            total_avg += crr[j];
        }
        total_avg = total_avg / M;
        printf("Total avg: %lf\n", total_avg);
    }

    MPI_Finalize();
    return 0;
}
```



```
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/HPCS LAB/LAB 3$ mpicc -o mpi_avg mpi_avg.c -lm
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/HPCS LAB/LAB 3$ mpirun -np 4 ./mpi_avg 3
Enter the array 4x3 :
1 2 3 4
5 6 7 8
9 10 11 12
Total avg: 5.000000
```

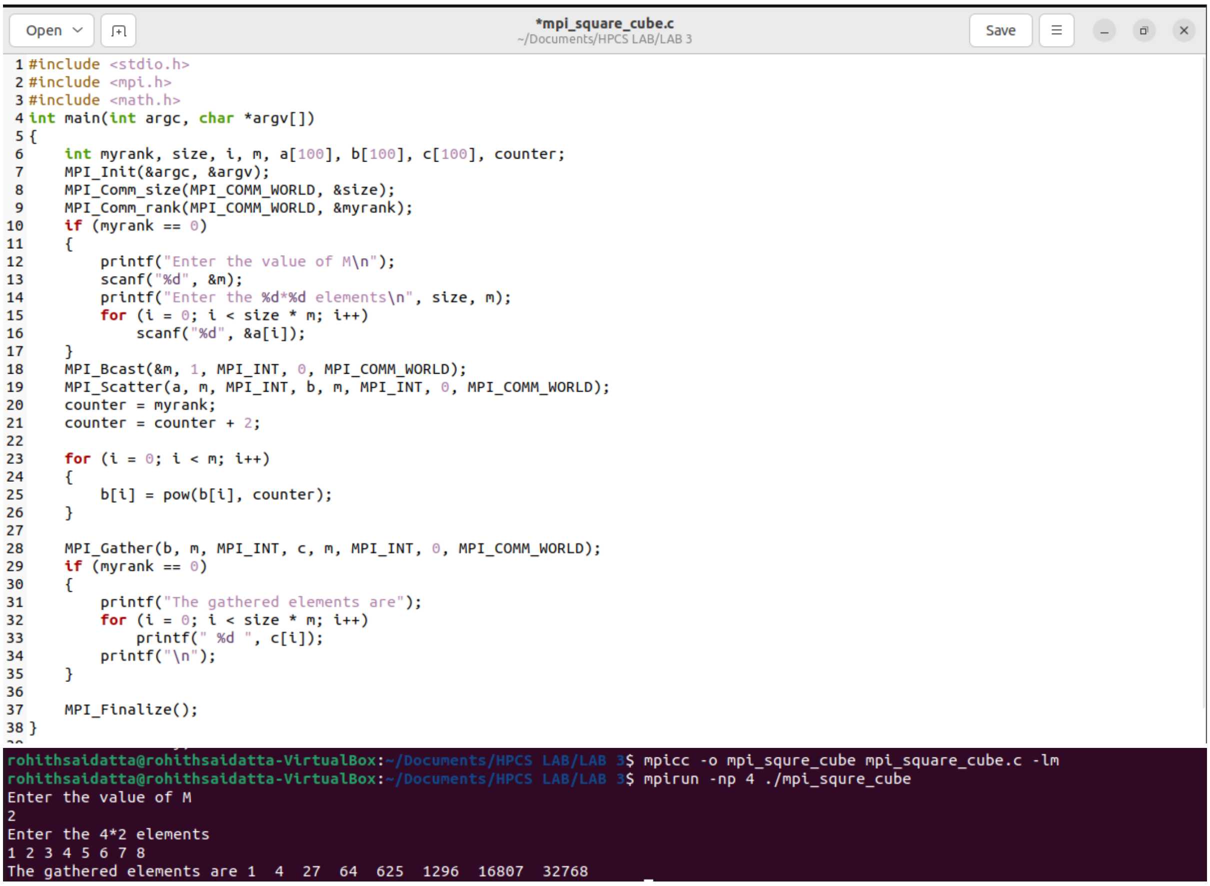
3) Write a MPI Program to read two strings S1 and S2 of same length in the root process. Using N process including the root (string length is evenly divisible by N), produce the concatenated resultant string as shown below. Display the resultant string in the root process.



```c
#include <stdio.h>
#include <mpi.h>
#include <string.h>
int main(int argc, char *argv[]) {
    int rank, size;
    int n, i, j;
    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Status status;

    char s1[100], s2[100], a[100], b[200];

    if (rank == 0) {
        printf("Enter the first string\n");
        scanf("%s", s1);
        printf("Enter the second string\n");
        scanf("%s", s2);
        n = strlen(s1);
    }

    MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
    MPI_Bcast(&s1, n, MPI_CHAR, 0, MPI_COMM_WORLD);
    MPI_Bcast(&s2, n, MPI_CHAR, 0, MPI_COMM_WORLD);

    for (i = 0, j = 0; i < n; i++, j += 2) {
        b[j] = s1[i];
        b[j + 1] = s2[i];
    }
    b[2 * n] = '\0';

    if (rank == 0) {
        printf("Resultant String: %s\n", b);
    }

    MPI_Finalize();
    return 0;
}
```

```
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/HPCS LAB/LAB 3$ mpicc mpi_string.c -o mpi_string
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/HPCS LAB/LAB 3$ mpirun -np 2 ./mpi_string
Enter the first string
hello
Enter the second string
world
Resultant String: hweolrllod
```

4) Write a program to read a value M and Nx M number of elements in the root. Using N processes do the following task. Find the square of first M numbers, Find the cube of next M numbers and so on. Print the results in the root.



```c
#include <stdio.h>
#include <mpi.h>
#include <math.h>
int main(int argc, char *argv[])
{
    int myrank, size, i, m, a[100], b[100], c[100], counter;
    MPI_Init(&argc, &argv);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Comm_rank(MPI_COMM_WORLD, &myrank);
    if (myrank == 0)
    {
        printf("Enter the value of M\n");
        scanf("%d", &m);
        printf("Enter the %d*%d elements\n", size, m);
        for (i = 0; i < size * m; i++)
            scanf("%d", &a[i]);
    }
    MPI_Bcast(&m, 1, MPI_INT, 0, MPI_COMM_WORLD);
    MPI_Scatter(a, m, MPI_INT, b, m, MPI_INT, 0, MPI_COMM_WORLD);
    counter = myrank;
    counter = counter + 2;

    for (i = 0; i < m; i++)
    {
        b[i] = pow(b[i], counter);
    }

    MPI_Gather(b, m, MPI_INT, c, m, MPI_INT, 0, MPI_COMM_WORLD);
    if (myrank == 0)
    {
        printf("The gathered elements are");
        for (i = 0; i < size * m; i++)
            printf(" %d ", c[i]);
        printf("\n");
    }

    MPI_Finalize();
}
```

```
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/HPCS LAB/LAB 3$ mpicc -o mpi_squre_cube mpi_square_cube.c -lm
rohithsaidatta@rohithsaidatta-VirtualBox:~/Documents/HPCS LAB/LAB 3$ mpirun -np 4 ./mpi_squre_cube
Enter the value of M
2
Enter the 4*2 elements
1 2 3 4 5 6 7 8
The gathered elements are 1  4  27  64  625  1296  16807  32768
```