Sahil Saini Salaria
Reg No. 180905048
Roll No. 11 C

<u>Solved</u>

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


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

   int arr[100],res[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 %d numbers:\n",size);
      for (int i = 0; i < size; i++)
      {
         scanf("%d",&arr[i]);
      }
   }

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

   ele=ele*ele;

   MPI_Gather(&ele,1,MPI_INT,res,1,MPI_INT,0,MPI_COMM_WORLD);

   if(rank==0)
   {
      for (int i = 0; i < size; i++)
      {
         printf("Rank %d , square of %d is: %d\n",i,arr[i],res[i]);
      }

   }
   MPI_Finalize();

   return 0;

}

//  mpicc solved.c -lm -o solved && mpirun -np 4 ./solved
```

```
student@selab-19:~/Desktop/sem6-Labs/PCAP Lab/lab3$ mpicc solved.c -lm -o solved && mpirun -np
4 ./solved
Enter the 4 numbers:
1
2
3
4
Rank 0 , square of 1 is: 1
Rank 1 , square of 2 is: 4
Rank 2 , square of 3 is: 9
Rank 3 , square of 4 is: 16
student@selab-19:~/Desktop/sem6-Labs/PCAP Lab/lab3$
```

<u>Exercise</u>

Q1

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


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

   int arr[100],res[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 %d numbers:\n",size);
      for (int i = 0; i < size; i++)
      {
         scanf("%d",&arr[i]);
      }
   }

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

   int fact=1;

   for (int i = 1; i <=ele; i++)
   {
      fact=fact*i;
   }


   MPI_Gather(&fact,1,MPI_INT,res,1,MPI_INT,0,MPI_COMM_WORLD);

   if(rank==0)
```

```
    {
        long long int result=0;
        for (int i = 0; i < size; i++)
        {
            printf("Rank %d, factorial of %d id : %d\n",i,arr[i],res[i]);
            result=result+res[i];
        }

        printf("Sum of all factorials is :%lld",result);

    }
    MPI_Finalize();

    return 0;

}

//  mpicc prog1.c -lm -o prog1 && mpirun -np 4 ./prog1
```

```
student@selab-19:~/Desktop/sem6-Labs/PCAP Lab/lab3$  mpicc prog1.c -lm -o prog1 && mpirun -np 4
 ./prog1
Enter the 4 numbers:
1
2
3
4
Rank 0, factorial of 1 id : 1
Rank 1, factorial of 2 id : 2
Rank 2, factorial of 3 id : 6
Rank 3, factorial of 4 id : 24
Sum of all factorials is :33student@selab-19:~/Desktop/sem6-Labs/PCAP Lab/lab3$
```

Q2

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


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

    int arr[1000],ele[100];
    float res[1000];

    float avg=0;

    MPI_Init(&argc,&argv);

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

```
    if(rank==0)
    {
        printf("Enter value of M:\n");
        scanf("%d",&M);

        printf("Enter the %d numbers:\n",size*M);
        for (int i = 0; i < size*M; i++)
        {
            scanf("%d",&arr[i]);
        }
    }
    MPI_Bcast(&M,1,MPI_INT,0,MPI_COMM_WORLD);
    MPI_Scatter(arr,M,MPI_INT,ele,M,MPI_INT,0,MPI_COMM_WORLD);

    for (int i = 0; i <M; i++)
    {
        avg=avg+ele[i];
    }
    avg=avg/M;


    MPI_Gather(&avg,1,MPI_FLOAT,res,1,MPI_FLOAT,0,MPI_COMM_WORLD);

    if(rank==0)
    {
        float final_avg=0;
        for (int i = 0; i < size; i++)
        {
            printf("Rank %d, avg is: %f\n",i,res[i]);
            final_avg=final_avg+res[i];
        }

        final_avg=final_avg/M;

        printf("Average is :%f",final_avg);

    }
    MPI_Finalize();

    return 0;

}

//  mpicc prog2.c -lm -o prog2 && mpirun -np 4 ./prog2
```

```
student@selab-19:~/Desktop/sem6-Labs/PCAP Lab/lab3$ mpicc prog2.c -lm -o prog2 && mpirun -np 4
./prog2
Enter value of M:
2
Enter the 8 numbers:
1
2
3
4
5
6
7
8
Rank 0, avg is: 1.500000
Rank 1, avg is: 3.500000
Rank 2, avg is: 5.500000
Rank 3, avg is: 7.500000
Average is :9.000000student@selab-19:~/Desktop/sem6-Labs/PCAP Lab/lab3$
```

Q3

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

int isConsonant(char ch)
{
   return !(ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u' || ch == 'A' || ch == 'E' || ch == 'I' || ch
== 'O' || ch == 'U');
}

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

   int slen;

   int send_size_per_process = 0;

   char str[1000], ele[1000];

   int res[100];

   int len = 0;

   MPI_Init(&argc, &argv);

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

   if (rank == 0)
   {
```

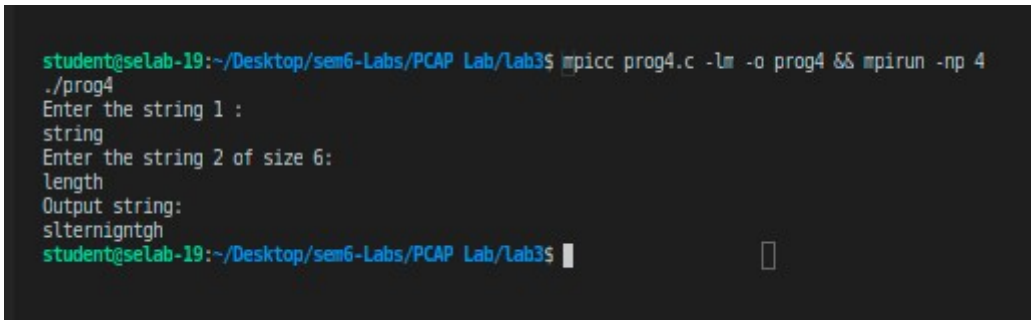
```
        printf("Enter the string\n");
        scanf("%s", str);

        len = strlen(str);

        send_size_per_process = (len - (len % size)) / size;
    }

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

    MPI_Scatter(str, send_size_per_process, MPI_CHAR, ele, send_size_per_process, MPI_CHAR,
0, MPI_COMM_WORLD);

    int tot_per_process = 0;

    for (int i = 0; i < send_size_per_process; i++)
    {
        if (isConsonant(ele[i]))
            tot_per_process++;
    }

    MPI_Gather(&tot_per_process, 1, MPI_INT, res, 1, MPI_INT, 0, MPI_COMM_WORLD);

    if (rank == 0)
    {
        int total = 0;
        for (int i = 0; i < size; i++)
        {
            printf("Rank %d, %d non-vowels\n", i, res[i]);
            total=total+res[i];
        }

        for (int i = size * send_size_per_process; i < len; i++)
        {
            if (isConsonant(str[i]))
            {
                total++;
            }
        }

        printf("Total non-vowels: %d ", total);
    }
    MPI_Finalize();

    return 0;
}

//  mpicc prog3.c -lm -o prog3 && mpirun -np 4 ./prog3
```

```
student@selab-19:~/Desktop/sem6-Labs/PCAP Lab/lab3$ mpicc prog3.c -lm -o prog3 && mpirun -np 4
./prog3
Enter the string
sahilsaini
Rank 0, 1 non-vowels
Rank 1, 1 non-vowels
Rank 2, 2 non-vowels
Rank 3, 0 non-vowels
Total non-vowels: 5 student@selab-19:~/Desktop/sem6-Labs/PCAP Lab/lab3$
```

Q4

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

int isConsonant(char ch)
{
   return !(ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u' || ch == 'A' || ch == 'E' || ch == 'I' || ch
== 'O' || ch == 'U');
}

int main(int argc, char *argv[])
{
   int rank, size;
   int slen;
   int send_size_per_process = 0;
   char str1[1000], str2[1000], ele1[1000], ele2[1000];
   char res[10000];
   int len = 0;

   MPI_Init(&argc, &argv);

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

   if (rank == 0)
   {
      printf("Enter the string 1 : \n");
      scanf("%s", str1);

      len = strlen(str1);

      printf("Enter the string 2 of size %d: \n", len);
      scanf("%s", str2);

      send_size_per_process = (len - (len % size)) / size;
   }

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

```
    MPI_Scatter(str1, send_size_per_process, MPI_CHAR, ele1, send_size_per_process,
MPI_CHAR, 0, MPI_COMM_WORLD);
    MPI_Scatter(str2, send_size_per_process, MPI_CHAR, ele2, send_size_per_process,
MPI_CHAR, 0, MPI_COMM_WORLD);

    char per_process_arr[1000];
    int count = 0;

    for (int i = 0; i < send_size_per_process; i++)
    {

            per_process_arr[count++] = ele1[i];
            per_process_arr[count++] = ele2[i];
    }

    MPI_Gather(per_process_arr, count, MPI_CHAR, res, count, MPI_CHAR, 0,
MPI_COMM_WORLD);

    if (rank == 0)
    {
        count = strlen(res);

        for (int i = count / 2; i < len; i++)
        {
                res[count++] = str1[i];
                res[count++] = str2[i];
        }
        printf("Output string: \n");
        puts(res);
    }
    MPI_Finalize();

    return 0;
}

//  mpicc prog4.c -lm -o prog4 && mpirun -np 4 ./prog4
```

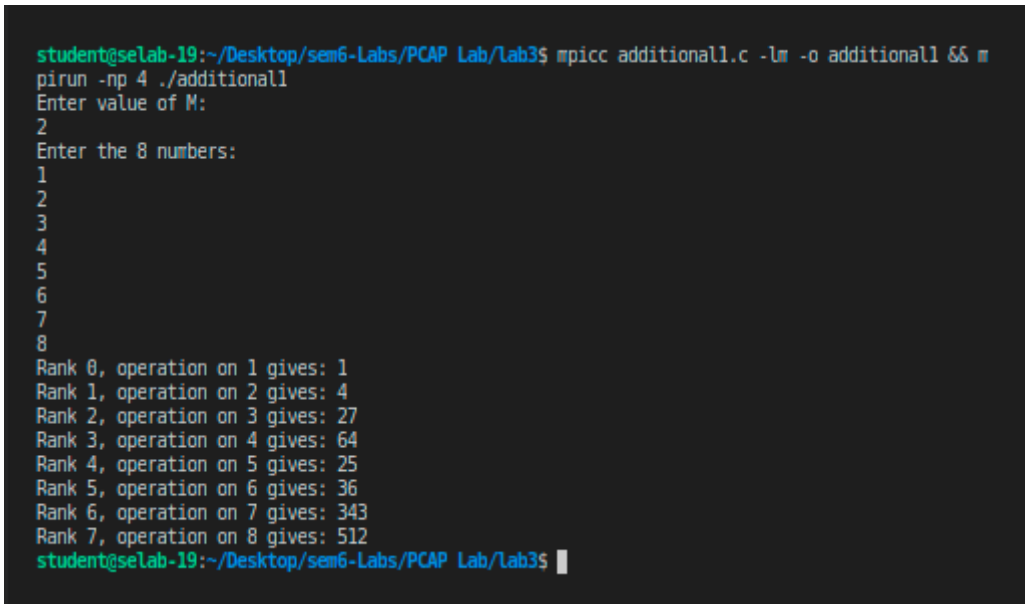


Additionals

Additional1

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

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

   int arr[1000], ele[1000];
   int res[1000];

   MPI_Init(&argc, &argv);

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

   if (rank == 0)
   {
      printf("Enter value of M:\n");
      scanf("%d", &M);

      printf("Enter the %d numbers:\n", size * M);
      for (int i = 0; i < size * M; i++)
      {
         scanf("%d", &arr[i]);
      }
   }
   MPI_Bcast(&M, 1, MPI_INT, 0, MPI_COMM_WORLD);
   MPI_Scatter(arr, M, MPI_INT, ele, M, MPI_INT, 0, MPI_COMM_WORLD);

   if (rank % 2 == 0)
   {
      for (int i = 0; i < M; i++)
      {
         ele[i] = ele[i] * ele[i];
      }
   }
   else
   {
      for (int i = 0; i < M; i++)
      {
         ele[i] = ele[i] * ele[i] * ele[i];
      }
   }

   MPI_Gather(ele, M, MPI_INT, res, M, MPI_INT, 0, MPI_COMM_WORLD);
```

```
    if (rank == 0)
    {
        for (int i = 0; i < size*M; i++)
        {
            printf("Rank %d, operation on %d gives: %d\n", i,arr[i], res[i]);
        }
    }
    MPI_Finalize();

    return 0;
}

//  mpicc additional1.c -lm -o additional1 && mpirun -np 4 ./additional1
```



```
student@selab-19:~/Desktop/sem6-Labs/PCAP Lab/lab3$ mpicc additional1.c -lm -o additional1 && m
pirun -np 4 ./additional1
Enter value of M:
2
Enter the 8 numbers:
1
2
3
4
5
6
7
8
Rank 0, operation on 1 gives: 1
Rank 1, operation on 2 gives: 4
Rank 2, operation on 3 gives: 27
Rank 3, operation on 4 gives: 64
Rank 4, operation on 5 gives: 25
Rank 5, operation on 6 gives: 36
Rank 6, operation on 7 gives: 343
Rank 7, operation on 8 gives: 512
student@selab-19:~/Desktop/sem6-Labs/PCAP Lab/lab3$
```

Additional2

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

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

    int arr[1000], ele[1000],res[1000];

    int total_e, total_o;

    MPI_Init(&argc, &argv);

    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
```

```
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    if (rank == 0)
    {
        printf("Enter value of M:\n");
        scanf("%d", &M);

        printf("Enter the %d numbers:\n", size * M);
        for (int i = 0; i < size * M; i++)
        {
            scanf("%d", &arr[i]);
        }
    }
    MPI_Bcast(&M, 1, MPI_INT, 0, MPI_COMM_WORLD);
    MPI_Scatter(arr, M, MPI_INT, ele, M, MPI_INT, 0, MPI_COMM_WORLD);

    int e_count = 0;
    int o_count = 0;
    for (int i = 0; i < M; i++)
    {
        if (ele[i] % 2 == 0)
            ele[i] = 1, e_count++;
        else
            ele[i] = 0, o_count++;
    }

    MPI_Gather(ele, M, MPI_INT, res, M, MPI_INT, 0, MPI_COMM_WORLD);
    MPI_Reduce(&e_count,&total_e,1 ,MPI_INT, MPI_SUM,0,MPI_COMM_WORLD);
    MPI_Reduce(&o_count,&total_o,1 ,MPI_INT, MPI_SUM,0,MPI_COMM_WORLD);


    if (rank == 0)
    {
        for (int i = 0; i < size * M; i++)
        {
            printf("Rank %d, operation on %d gives: %d\n", i, arr[i], res[i]);
        }

        printf("Taotal even numbers in array are:%d \n",total_e);
        printf("Taotal odd numbers in array are:%d \n",total_o);

    }
    MPI_Finalize();

    return 0;
}

// https://cvw.cac.cornell.edu/mpicc/scan

//  mpicc additional2.c -lm -o additional2 && mpirun -np 4 ./additional2
```

```
student@selab-19:~/Desktop/sem6-Labs/PCAP Lab/lab3$ mpicc additional2.c -lm -o additional2 && m
pirun -np 4 ./additional2
Enter value of M:
2
Enter the 8 numbers:
1
2
3
4
5
6
7
8
Rank 0, operation on 1 gives: 0
Rank 1, operation on 2 gives: 1
Rank 2, operation on 3 gives: 0
Rank 3, operation on 4 gives: 1
Rank 4, operation on 5 gives: 0
Rank 5, operation on 6 gives: 1
Rank 6, operation on 7 gives: 0
Rank 7, operation on 8 gives: 1
Taotal even numbers in array are:4
Taotal odd numbers in array are:4
student@selab-19:~/Desktop/sem6-Labs/PCAP Lab/lab3$
```

//ENDS//