Lab No 3: Date: 21/01/2026

Collective Communications in MPI

Name: Satyam Singh

Reg no: 230905256

CSE B2 - 37

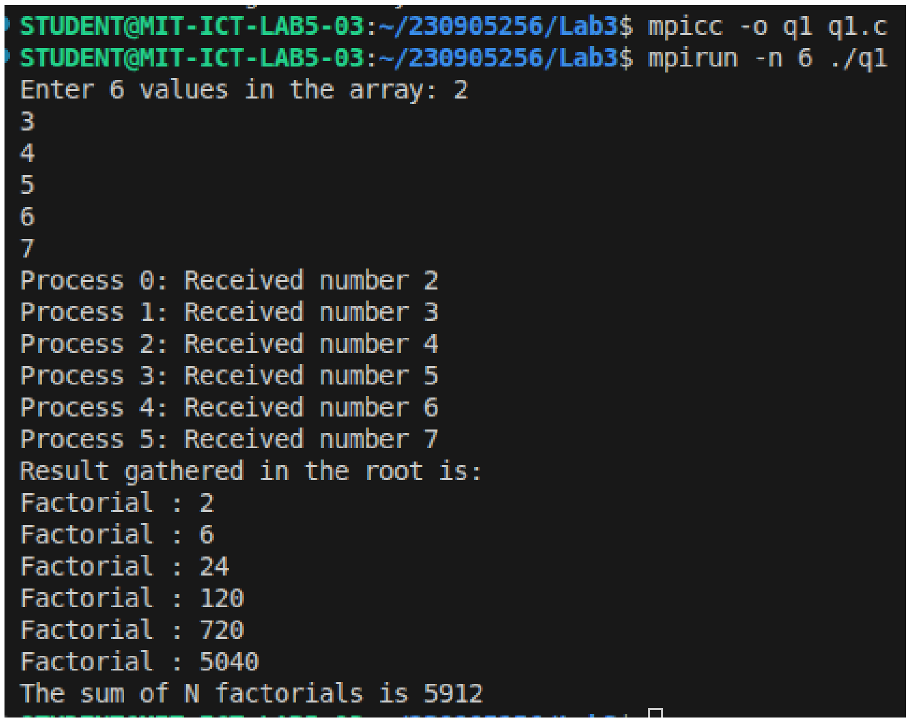
Lab Exercises:

Q1) Write a MPI program to read N values in the root process. Root process sends one value to each process. Every process receives it and finds the factorial and finds sum of it. Use N number of processes.

Ans)

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

int fact(int n){
if(n==0||n==1) return 1;
return n*fact(n-1);
}

int main(int argc, char *argv[]){
int rank, size, n, a[10], b[10],c;
MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &size);

if(rank == 0){
n = size;
printf("Enter %d values in the array: ", size);
for(int i = 0; i < n; i++) scanf("%d", &a[i]);
}
MPI_Scatter(a,1,MPI_INT,&c,1,MPI_INT,0,MPI_COMM_WORLD);
printf("Process %d: Received number %d\n", rank, c);
c = fact(c);
MPI_Gather(&c,1,MPI_INT,b,1,MPI_INT,0,MPI_COMM_WORLD);

if(rank == 0){
int sum = 0;
printf("Result gathered in the root is: \n");
for(int i =0 ; i <n; i++) {
printf("Factorial : %d\n", b[i]);
sum += b[i];
}
printf("The sum of N factorials is %d\n", sum);
}

MPI_Finalize();
return 0;
}
```

Output:



```
STUDENT@MIT-ICT-LAB5-03:~/230905256/Lab3$ mpicc -o q1 q1.c
STUDENT@MIT-ICT-LAB5-03:~/230905256/Lab3$ mpirun -n 6 ./q1
Enter 6 values in the array: 2
3
4
5
6
7
Process 0: Received number 2
Process 1: Received number 3
Process 2: Received number 4
Process 3: Received number 5
Process 4: Received number 6
Process 5: Received number 7
Result gathered in the root is:
Factorial : 2
Factorial : 6
Factorial : 24
Factorial : 120
Factorial : 720
Factorial : 5040
The sum of N factorials is 5912
```

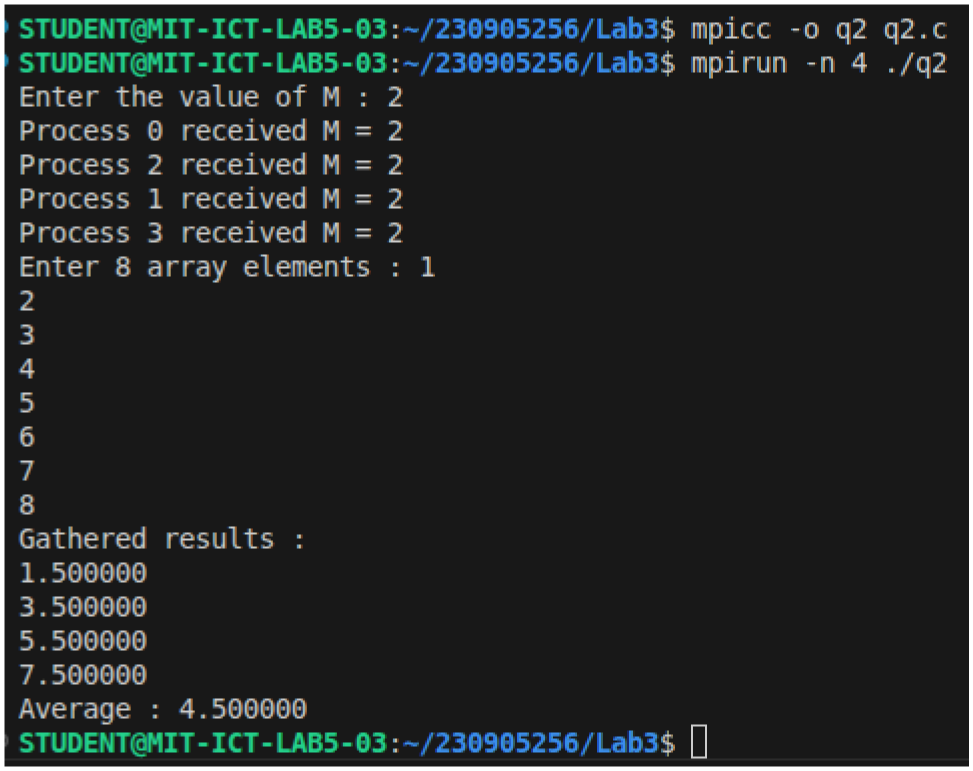
Q2) Write a MPI program to read value M and N * M elements into an 1D array in the root process, where N is the number of processes. Root process sends M elements to each process. Each process finds average of M elements it receuved and sends these average values to root. Root collects all the values and finds the total average. Use collective communication routines.

Ans)

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

int main(int argc, char *argv[]){
int rank, size, N, M;
MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &size);
N = size;

if(rank==0){
printf("Enter the value of M : ");
scanf("%d", &M);
}
MPI_Bcast(&M, 1, MPI_INT, 0, MPI_COMM_WORLD);
printf("Process %d received M = %d\n", rank, M);
int arr_size = N * M;
int A[arr_size], rcvbuf[M];
float averages[N];

if (rank==0){
sleep(1);
printf("Enter %d array elements : ", arr_size);
for(int i=0;i<arr_size;i++) scanf("%d",&A[i]);
}

MPI_Scatter(A,M,MPI_INT,&rcvbuf,M,MPI_INT,0,MPI_COMM_WORLD);

float avg=0.0;
for (int i=0;i<M;i++){
```

```
avg += rcvbuf[i];
}
avg = avg/M;

MPI_Gather(&avg,1,MPI_INT,averages,1,MPI_INT,0,MPI_COMM_WORLD);

if(rank==0){
printf("Gathered results : \n");
avg = 0.0;
for(int i=0;i<N;i++){
printf("%f \n",averages[i]);
avg += averages[i];
}
avg = avg/N;
printf("Average : %f\n",avg);
}

MPI_Finalize();
return 0;
}
```

Output:



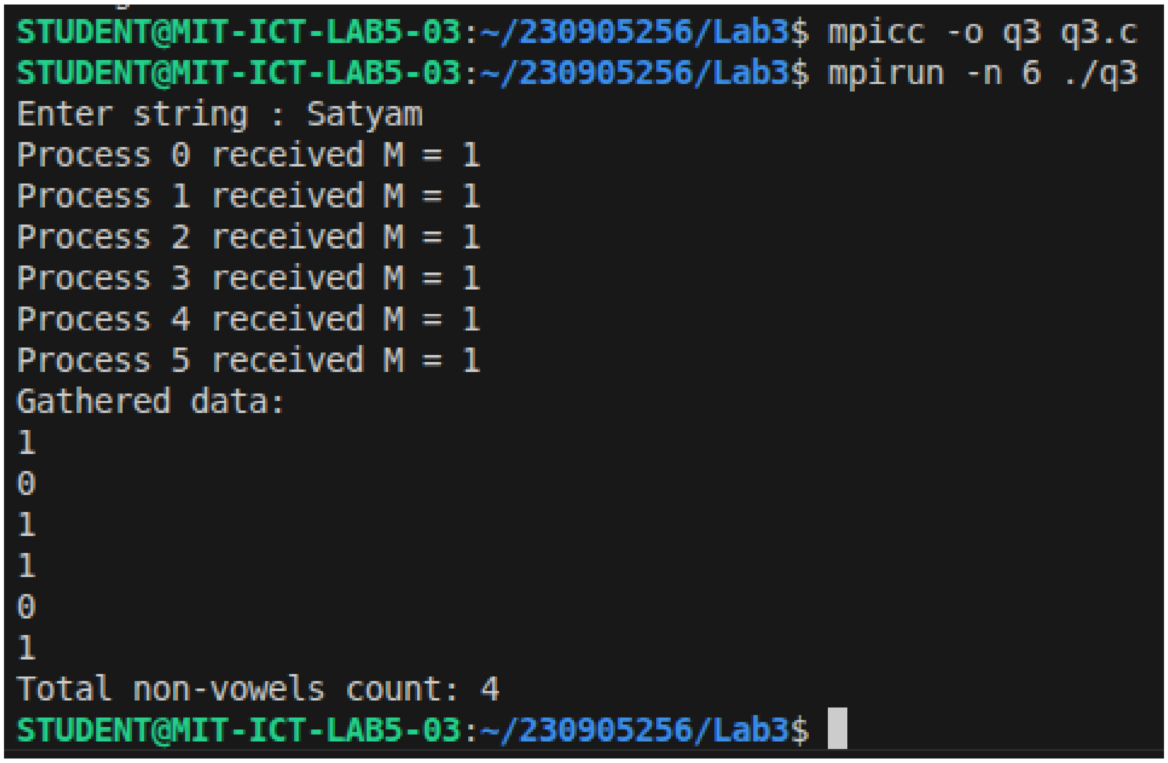
```
STUDENT@MIT-ICT-LAB5-03:~/230905256/Lab3$ mpicc -o q2 q2.c
STUDENT@MIT-ICT-LAB5-03:~/230905256/Lab3$ mpirun -n 4 ./q2
Enter the value of M : 2
Process 0 received M = 2
Process 2 received M = 2
Process 1 received M = 2
Process 3 received M = 2
Enter 8 array elements : 1
2
3
4
5
6
7
8
Gathered results :
1.500000
3.500000
5.500000
7.500000
Average : 4.500000
STUDENT@MIT-ICT-LAB5-03:~/230905256/Lab3$ 
```

Q3) Write a MPI program to read a string. Using N processes(string length is evenly divisible by N), find the number of non-vowels in the string. In the root process print number of non-vowels found by each process and print the total number of non-vowels.

Ans)

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

int main(int argc, char *argv[]){
int rank, size, N, M, total_vowels = 0;
MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &size);
N = size;
char string[100];
int process_vowels[N];

if(rank==0){
printf("Enter string : ");
scanf("%[^\n]c", string);
M = strlen(string) / N;
}

MPI_Bcast(&M, 1, MPI_INT, 0, MPI_COMM_WORLD);
printf("Process %d received M = %d\n", rank, M);
char rcvbuf[M];
int num_vowels = 0;
char vowels[10] = {'a', 'e', 'i', 'o', 'u','A', 'E', 'I', 'O', 'U'};

MPI_Scatter(string, M, MPI_CHAR, rcvbuf, M, MPI_CHAR, 0, MPI_COMM_WORLD);
int flag = 1;
for (int i=0;i<M;i++){
for (int j=0;j<10;j++){
if (rcvbuf[i] == vowels[j])
flag = 0;
}
if (flag==1) num_vowels++;
flag=1;
}

MPI_Gather(&num_vowels, 1, MPI_INT, &process_vowels, 1, MPI_INT, 0, MPI_COMM_WORLD);

if (rank==0){
sleep(1);
printf("Gathered data: \n");
for (int i=0;i<N;i++){
printf("%d\n", process_vowels[i]);
total_vowels +=process_vowels[i];
}
printf("Total non-vowels count: %d\n", total_vowels);
}

MPI_Finalize();
return 0;
}
```

Output:



```
STUDENT@MIT-ICT-LAB5-03:~/230905256/Lab3$ mpicc -o q3 q3.c
STUDENT@MIT-ICT-LAB5-03:~/230905256/Lab3$ mpirun -n 6 ./q3
Enter string : Satyam
Process 0 received M = 1
Process 1 received M = 1
Process 2 received M = 1
Process 3 received M = 1
Process 4 received M = 1
Process 5 received M = 1
Gathered data:
1
0
1
1
0
1
Total non-vowels count: 4
STUDENT@MIT-ICT-LAB5-03:~/230905256/Lab3$
```

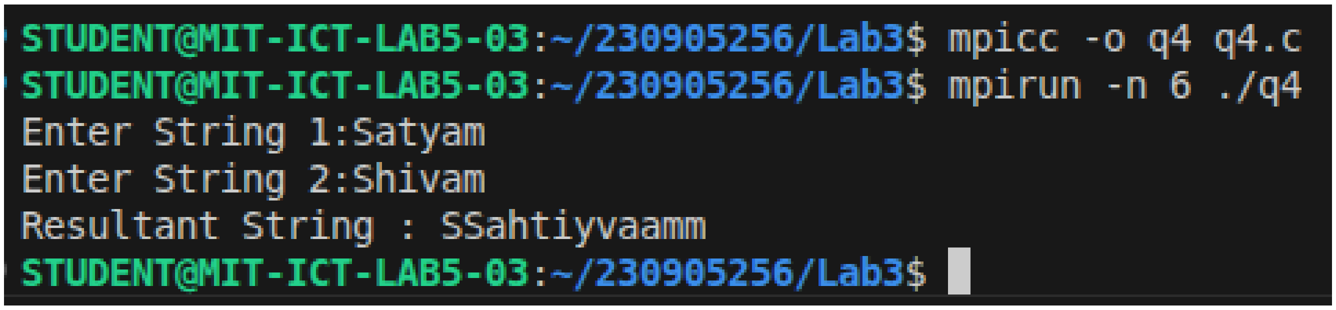
Q4) Write a MPI program to read two strings S1 and S2 of same length in the root process. Using N processes including the root (string length is evenly divisible by N), produce the resultant string as shown below. Display the resultant string in the root process. Use collective communication routines.

Ans)

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

int main(int argc,char** argv){
int rank,size,N,i,M,l=0;
char str1[100];
char str2[100];
char B1[100];
char C[200];
char concat[100];
MPI_Init(&argc,&argv);
MPI_Comm_rank(MPI_COMM_WORLD,&rank);
MPI_Comm_size(MPI_COMM_WORLD,&size);
if(rank==0){
N = size;
printf("Enter String 1:");
scanf("%[^\n]c",str1);
printf("Enter String 2:");
scanf(" %[^\n]c",str2);
M = strlen(str1)/N;
}
MPI_Bcast(&M,1,MPI_INT,0,MPI_COMM_WORLD);
MPI_Scatter(str1,M,MPI_CHAR,B1,M,MPI_CHAR,0,MPI_COMM_WORLD);
MPI_Scatter(str2,M,MPI_CHAR,B1+M,M,MPI_CHAR,0,MPI_COMM_WORLD);
l=0;
for(i=0;i<M;i++){
concat[l++] = B1[i];
concat[l++] = B1[i+M];
}
MPI_Gather(concat,2*M,MPI_CHAR,C,2*M,MPI_CHAR,0,MPI_COMM_WORLD);
if(rank==0){
```

```
printf("Resultant String : %s\n",C);
}
MPI_Finalize();
return 0;
}
```

Output:



```
STUDENT@MIT-ICT-LAB5-03:~/230905256/Lab3$ mpicc -o q4 q4.c
STUDENT@MIT-ICT-LAB5-03:~/230905256/Lab3$ mpirun -n 6 ./q4
Enter String 1:Satyam
Enter String 2:Shivam
Resultant String : SSahtiyvaamm
STUDENT@MIT-ICT-LAB5-03:~/230905256/Lab3$
```