Name : VIVEK SINGH
RegNumber : 230905408
Roll Number: A50

Lab 3: Collective Communications in MPI

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 of that number and returns it to the root process. Root process gathers the factorial and finds sum of it. Use N number of processes.

Code:

```
#include <mpi.h>
#include <stdio.h>
int fact(int x)
{
	int f = 1;
	for(int i = 1; i <= x; i++)
		f *= i;
	return f;
}
int main(int argc, char *argv[])
{
	int rank, size, A[10], B[10], x, sum = 0;
	MPI_Init(&argc, &argv);
	MPI_Comm_rank(MPI_COMM_WORLD, &rank);
	MPI_Comm_size(MPI_COMM_WORLD, &size);
	if(rank == 0){
		printf("Enter %d values:\n", size);
		for(int i = 0; i < size; i++)
			scanf("%d", &A[i]);
	}
	MPI_Scatter(A, 1, MPI_INT, &x, 1, MPI_INT, 0, MPI_COMM_WORLD);
	x = fact(x);
	MPI_Gather(&x, 1, MPI_INT, B, 1, MPI_INT, 0, MPI_COMM_WORLD);
	if(rank == 0){
		for(int i = 0; i < size; i++)
			sum += B[i];
	printf("Sum of factorials = %d\n", sum);
	}
	MPI_Finalize();
	return 0;
}
```

Output :

```
STUDENT@MIT-ICT-LAB5-15:~/Desktop/230905408/lab3$ mpirun -n 4 ./Q1
Enter 4 values:
1 2 3 4
Sum of factorials = 33
STUDENT@MIT-ICT-LAB5-15:~/Desktop/230905408/lab3$
```

Q2)Write a MPI program to read an integer value M and NXM elements into an ID 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 received and sends these average values to root. Root collects all the values and finds the total average. Use collective com- munication routines.

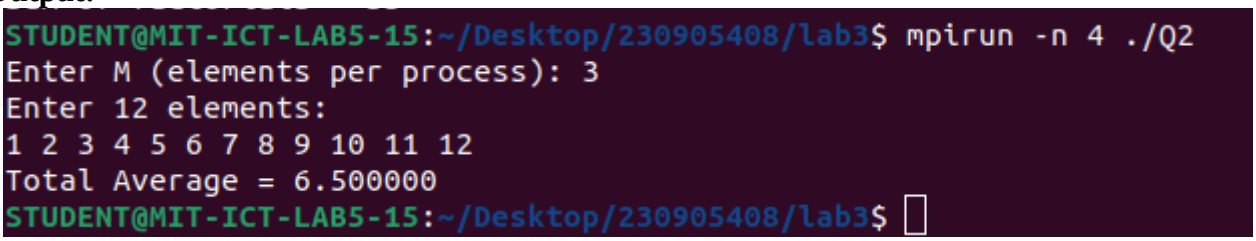
Code:

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

int main(int argc, char *argv[])
{
	int rank, size, M;
	float A[100], sub[20], avg, avg_all[10], total = 0;
	MPI_Init(&argc, &argv);
	MPI_Comm_rank(MPI_COMM_WORLD, &rank);
	MPI_Comm_size(MPI_COMM_WORLD, &size);
	if(rank == 0){
		printf("Enter M (elements per process): ");
		scanf("%d", &M);
		printf("Enter %d elements:\n", M * size);
		for(int i = 0; i < M * size; i++)
			scanf("%f", &A[i]);
	}
	MPI_Bcast(&M, 1, MPI_INT, 0, MPI_COMM_WORLD);
	MPI_Scatter(A, M, MPI_FLOAT, sub, M, MPI_FLOAT, 0, MPI_COMM_WORLD);
	avg = 0;
	for(int i = 0; i < M; i++)
		avg += sub[i];
		avg /= M;
	MPI_Gather(&avg, 1, MPI_FLOAT, avg_all, 1, MPI_FLOAT, 0, MPI_COMM_WORLD);
	if(rank == 0){
		for(int i = 0; i < size; i++)
			total += avg_all[i];
		printf("Total Average = %f\n", total / size);
	}
	MPI_Finalize();
```

```
    return 0;
}
```

Output:



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.

Code:

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

int isVowel(char c){
    return (c=='a'||c=='e'||c=='i'||c=='o'||c=='u'||
    c=='A'||c=='E'||c=='I'||c=='O'||c=='U');
}
int main(int argc, char *argv[]){
    int rank, size, len;
    int count = 0, counts[10];
    char str[100], sub[20];
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    if(rank == 0){
        printf("Enter string: ");
        scanf("%s", str);
        len = strlen(str);
    }
    MPI_Bcast(&len, 1, MPI_INT, 0, MPI_COMM_WORLD);
    MPI_Scatter(str, len/size, MPI_CHAR,sub, len/size, MPI_CHAR, 0, MPI_COMM_WORLD);
    for(int i = 0; i < len/size; i++)
        if(!isVowel(sub[i]))    count++;
    MPI_Gather(&count, 1, MPI_INT, counts, 1, MPI_INT, 0, MPI_COMM_WORLD);
    if(rank == 0){
        int total = 0;
        for(int i = 0; i < size; i++)          total += counts[i];
        printf("Total non-vowels = %d\n", total);
    }
    MPI_Finalize();
}
```

Output:

```
STUDENT@MIT-ICT-LAB5-15:~/Desktop/230905408/lab3$ mpirun -n 4 ./Q3
Enter string: hello
Total non-vowels = 3
STUDENT@MIT-ICT-LAB5-15:~/Desktop/230905408/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.
Example:
String S1: string
String S2: length
Resultant String: slternigntgh

Code:

```
#include <mpi.h>
#include <stdio.h>
#include <string.h>
int main(int argc, char *argv[])
{
	int rank, size, len;
	char s1[100], s2[100];
	char r1[20], r2[20], sub[40], res[200];
	MPI_Init(&argc, &argv);
	MPI_Comm_rank(MPI_COMM_WORLD, &rank);
	MPI_Comm_size(MPI_COMM_WORLD, &size);
	if(rank == 0){
		printf("Enter string S1: ");
		scanf("%s", s1);
		printf("Enter string S2: ");
		scanf("%s", s2);
		len = strlen(s1);
	}
	MPI_Bcast(&len, 1, MPI_INT, 0, MPI_COMM_WORLD);
	MPI_Scatter(s1, len/size, MPI_CHAR, r1, len/size, MPI_CHAR, 0, MPI_COMM_WORLD);
	MPI_Scatter(s2, len/size, MPI_CHAR, r2, len/size, MPI_CHAR, 0, MPI_COMM_WORLD);
	int k = 0;
	for(int i = 0; i < len/size; i++){
		sub[k++] = r1[i];
		sub[k++] = r2[i];
	}
	MPI_Gather(sub, 2*(len/size), MPI_CHAR,res, 2*(len/size), MPI_CHAR, 0,
		MPI_COMM_WORLD);
```

```
        if(rank == 0){
                res[2*len] = '\0';
                printf("Resultant string = %s\n", res);
        }
        MPI_Finalize();
        return 0;
}
```

Output: