Dennis Johnson
180905025 Lab3 B1

Q1.

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

int factorial(int n){
	if(n <= 2)
		return n;

	return n * factorial(n-1);
}

int main(int argc, char  *argv[]){
	int rank, size, N, A[10], B[10], c, i;

	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\n", N);
		fflush(stdout);
		for(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 %d\n", rank, c);
	fflush(stdout);

	c = factorial(c);
	printf("Process %d computed %d\n", rank, c);

	MPI_Gather(&c, 1, MPI_INT, B, 1, MPI_INT, 0, MPI_COMM_WORLD);

	if(rank == 0){
		printf("Result gathered in root\n");
		fflush(stdout);
		for(i = 0; i < N; i++)
			printf("%d\t", B[i]);

		printf("\n");
		fflush(stdout);
	}

	MPI_Finalize();
	return 0;
}
```

```
student@lplab-Lenovo-Product:~/180905025_PP/lab3$ mpirun -np 4 ./a.out
Enter 4 values
1
2
3
4
Process 0 received 1
Process 0 computed 1
Result gathered in root
1        2        6        24
Process 1 received 2
Process 1 computed 2
Process 2 received 3
Process 2 computed 6
Process 3 received 4
Process 3 computed 24
```

Q2.

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

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

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

        int N = size, M = 0;
        float collected[N], indv_avg = 0;
        int *Arr;

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

                Arr = calloc(N*M, sizeof(int));

                printf("Enter %d values\n", N*M);
                fflush(stdout);
                for(int i = 0; i < M*N; i++)
                        scanf("%d", &Arr[i]);
        }

        int recv[M];
        MPI_Bcast(&M, 1, MPI_INT, 0, MPI_COMM_WORLD);
        MPI_Scatter(Arr, M, MPI_INT, recv, M, MPI_INT, 0, MPI_COMM_WORLD);
        fflush(stdout);
```

```
	//Compute individual average
	for(int i = 0; i < M; i++){
		indv_avg += recv[i];
	}
	indv_avg /= M;

	printf("Process %d computed indv_average %f\n", rank, indv_avg);

	MPI_Gather(&indv_avg, 1, MPI_FLOAT, collected, 1, MPI_FLOAT, 0,
MPI_COMM_WORLD);

	if(rank == 0){
		float global_avg = 0;

		fflush(stdout);
		for(int i = 0; i < N; i++){
			//printf("%f\t", collected[i]);
			global_avg += collected[i];
		}
		global_avg /= N;

		printf("Global Average is %f\n", global_avg);
		fflush(stdout);
	}

	MPI_Finalize();
	return 0;
}
```

```
student@lplab-Lenovo-Product:~/180905025_PP/lab3$ mpirun -np 4 ./a.out
Enter value for M: 2
Enter 8 values
1
2
3
4
5
6
7
8
Process 0 computed indv_average 1.500000
Global Average is 4.500000
Process 1 computed indv_average 3.500000
Process 2 computed indv_average 5.500000
Process 3 computed indv_average 7.500000
student@lplab-Lenovo-Product:~/180905025_PP/lab3$
```

Q3.

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

int isVowel(char ch){
	if(ch =='a'||ch=='e'||ch=='i'||ch=='o'||ch=='u')
		return 1;
	return 0;
}

int main(int argc, char  *argv[]){
	int rank, size, str_len = 0, indv_count = 0;

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

	int N = size, len_each = 0;
	char str[256];

	if(rank == 0){
		printf("Enter contents of string\n");
		fflush(stdout);
		gets(str);
		str_len = strlen(str);

		if(str_len % N != 0){
			fprintf(stderr, "String length is not divisible by N\n");
			exit(1);
		}

		len_each = str_len / N;
	}

	char recv[len_each];
	int collected[N];

	MPI_Bcast(&len_each, 1, MPI_INT, 0, MPI_COMM_WORLD);
	MPI_Scatter(str, len_each, MPI_CHAR, recv, len_each, MPI_CHAR, 0,
MPI_COMM_WORLD);
	fflush(stdout);

	printf("Process %d received %s\n", rank, recv);
	for(int i=0; i <len_each; i++){
		if(isVowel(recv[i]))
```

```
                indv_count++;
        }

        MPI_Gather(&indv_count, 1, MPI_INT, collected, 1, MPI_INT, 0, MPI_COMM_WORLD);

        if(rank == 0){
                int total_count = 0;

                fflush(stdout);
                for(int i = 0; i < N; i++){
                        total_count += collected[i];
                }

                printf("Total vowel count is %d\n", total_count);
                fflush(stdout);
        }

        MPI_Finalize();
        return 0;
}
```

```
student@lplab-Lenovo-Product:~/180905025_PP/lab3$ mpirun -np 4 ./a.out
Enter contents of string
dennisjj
Process 0 received de
Process 1 received nn
Process 2 received is
Process 3 received jj
Total vowel count is 2
student@lplab-Lenovo-Product:~/180905025_PP/lab3$
```

Q4.

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


int main(int argc, char  *argv[]){
        int rank, size, str_len = 0, indv_count = 0;
```

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

	int N = size, len_each = 0;
	char str1[256], str2[256];

	if(rank == 0){
		printf("Enter contents of string1\n");
		fflush(stdout);
		gets(str1);

		printf("Enter contents of string1\n");
		gets(str2);
		fflush(stdout);

		str_len = strlen(str1);

		if(str_len % N != 0 || strlen(str2) != str_len){
			fprintf(stderr, "String length is not divisible by N\n");
			exit(1);
		}

		len_each = str_len / N;
	}


	char *indv_cat;
	char collected[256];

	MPI_Bcast(&len_each, 1, MPI_INT, 0, MPI_COMM_WORLD);
	char recv1[len_each], recv2[len_each];
	MPI_Scatter(str1, len_each, MPI_CHAR, recv1, len_each, MPI_CHAR, 0,
MPI_COMM_WORLD);
	MPI_Scatter(str2, len_each, MPI_CHAR, recv2, len_each, MPI_CHAR, 0,
MPI_COMM_WORLD);
	fflush(stdout);

	printf("Process %d received %s and %s\n", rank, recv1, recv2);
	indv_cat = strcat(recv1, recv2);
	printf("Process %d concat %s\n", rank, indv_cat);

	MPI_Gather(indv_cat, strlen(indv_cat)+1, MPI_CHAR, collected, len_each*2, MPI_CHAR,
0, MPI_COMM_WORLD);

	if(rank == 0){
		char final[256];
		fflush(stdout);
		printf("Recevied at root %s\n", collected);
	}

	MPI_Finalize();
```

```
        return 0;
}
```

```
student@lplab-Lenovo-Product:~/180905025_PP/lab3$ mpirun -np 4 ./a.out
Enter contents of string1
stringaa
Enter contents of string1
lengthbb
```