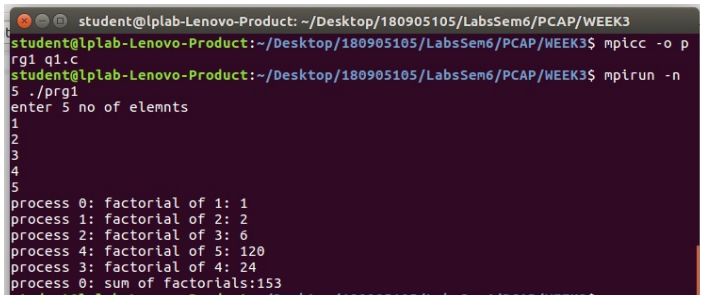
LAB 3
HRITIK AJAY BANSAL
CSE A 15
180905105

Q1)

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

int facto(int c){
	int t=1;int flag=1;
	for (int i = 1; i <=c; ++i)
	{
		t*=i;
	}
	return t;
}
int main(int argc, char const *argv[])
{
	int fact;
	int x;
	int rank;int size;
	int root=0;int flag=1;

	MPI_Init(NULL,NULL);
	MPI_Comm_rank(MPI_COMM_WORLD,&rank);
	MPI_Comm_size(MPI_COMM_WORLD,&size);
	int a[size],b[size];
	if(rank==root){
		printf("enter %d no of elemnts\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);

	fact=facto(x);
	printf("process %d: factorial of %d: %d\n",rank,x,fact );

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

	if (rank==0)
	{	int recv=0;
		for (int i = 0; i < size; ++i)
		{
			recv+=b[i];
		}
		printf("process 0: sum of factorials:%d\n",recv );
	}
```

```
	MPI_Finalize();
	return 0;
}
```

OUTPUT:



Q2)

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

double avrg(int* b,int m){
	double sum=0,avv;
	for (int i = 0; i < m; ++i)
	{
		sum+=b[i];
	}
	avv=sum/m;
	return avv;
}
int main(int argc, char const *argv[])
{
	int rank;int size;
	int root=0;int flag=1;

	MPI_Init(NULL,NULL);
	MPI_Comm_rank(MPI_COMM_WORLD,&rank);
	MPI_Comm_size(MPI_COMM_WORLD,&size);\
	int n=size;
	int m;
	int *a;
	double avr[n];
	if (rank==root)
	{
		printf("enter the value of m\n");
		scanf("%d",&m);
		a=malloc(m*n*(sizeof(int)));
		printf("enter %d elements\n",n*m);
		for (int i = 0; i < n*m; ++i)
		{
			scanf("%d",&a[i]);
		}

	}
```

```
	MPI_Bcast(&m,1,MPI_INT,0,MPI_COMM_WORLD);
	int *b;
	b=malloc(m*sizeof(int));

	MPI_Scatter(a,m,MPI_INT,b,m,MPI_INT,0,MPI_COMM_WORLD);
	double avg=avrg(b,m);
	printf("process %d: average of my m nos is: %f\n",rank,avg);

	MPI_Gather(&avg,1,MPI_DOUBLE,avr,1,MPI_DOUBLE,0,MPI_COMM_WORLD);
	if (rank==0)
	{double sum=0;
		for (int i = 0; i < n; ++i)
		{
			sum+=avr[i];

		}
		double avg1=((sum)/n);
		printf("process 0: average of all nos is: %f\n",avg1);
	}

	MPI_Finalize();
	return 0;
}
```

output:

Q3)

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

int non_count(char* b){
	int non=0;
	for (int i = 0; i < strlen(b); ++i)
	{
		if(isalpha(b[i]))
		if (b[i]!='a'&&b[i]!='e'&&b[i]!='i'&&b[i]!='o'&&b[i]!='u'&&b[i])
		{
			non++;
		}
	}
```

```
        return non;
}
int main(int argc, char const *argv[])
{
        int fact;
        int x;
        int rank;int size;
        int root=0;int flag=1;

        MPI_Init(NULL,NULL);
        MPI_Comm_rank(MPI_COMM_WORLD,&rank);
        MPI_Comm_size(MPI_COMM_WORLD,&size);
        char a[100],b[100];
        int c[size];
        int m;
        if(rank==root){
                printf("enter a string having multiple of %d characters\n",size );
                gets(a);
                m=(strlen(a)/size);


        }
        MPI_Bcast(&m,1,MPI_INT,0,MPI_COMM_WORLD);

        MPI_Scatter(a,m,MPI_CHAR,&b,m,MPI_CHAR,0,MPI_COMM_WORLD);

        int nonc=non_count(b);
        printf("process %d: non-vowel alphabet count is: %d\n",rank,nonc );

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

        if (rank==0)
        {       int recv=0;
                for (int i = 0; i < size; ++i)
                {
                        recv+=c[i];
                }
                printf("process %d: final non-vowel alphabet count is: %d\n",rank,recv );
        }
        MPI_Finalize();
        return 0;
}
```

output:

```
student@lplab-Lenovo-Product:~/Desktop/180905105/LabsSem6/PCAP/WEEK3$ mpicc -o prg3 q3.c
q3.c: In function ‘non_count’:
q3.c:9:6: warning: implicit declaration of function ‘isalpha’ [-Wimplicit-function-declaration]
   if(isalpha(b[i]))
      ^
q3.c: In function ‘main’:
q3.c:32:3: warning: implicit declaration of function ‘gets’ [-Wimplicit-function-declaration]
   gets(a);
   ^
/tmp/ccYzOtrO.o: In function `main':
q3.c:(.text+0x205): warning: the `gets' function is dangerous and should not be used.
student@lplab-Lenovo-Product:~/Desktop/180905105/LabsSem6/PCAP/WEEK3$ mpirun -n 3 ./prg3
enter a string having multiple of 3 characters
hritik is my
process 0: non-vowel alphabet count is: 3
process 0: final non-vowel alphabet count is: 7
process 1: non-vowel alphabet count is: 1
process 2: non-vowel alphabet count is: 3
student@lplab-Lenovo-Product:~/Desktop/180905105/LabsSem6/PCAP/WEEK3$
```

Q4)

```
#include "mpi.h"
#include <stdio.h>
#include <string.h>
int main(int argc, char* argv [] ) {
int rank, size;
float avg = 0;
char b[100], str1[100], str2[100], c1[100], c2[100], concatted[100];
int i, j, m;
MPI_Init(&argc, &argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);MPI_Comm_size(MPI_COMM_WORLD,
&size);
if (rank == 0) {
printf("Enter String 1\n");
scanf("%s",str1);
printf("Enter String 2\n");
scanf("%s",str2);
m = strlen(str1) / size;
}
MPI_Bcast(&m, 1, MPI_INT, 0, MPI_COMM_WORLD);
MPI_Scatter(str1, m, MPI_CHAR, c1, m, MPI_CHAR, 0, MPI_COMM_WORLD);
MPI_Scatter(str2, m, MPI_CHAR, c2, m, MPI_CHAR, 0, MPI_COMM_WORLD);
int t = 0;
for (t = 0; t <= 2 * m; t += 2) {
concatted[t] = c1[t/2];
concatted[t+1] = c2[t/2];
}
concatted[2*m] = '\0';
MPI_Gather(concatted, 2*m, MPI_CHAR, b, 2*m, MPI_CHAR, 0,
MPI_COMM_WORLD);
if (rank == 0) {
b[m*size*2] = '\0';
printf("Concatted: %s\n",b);
}
MPI_Finalize();
}
```

output: