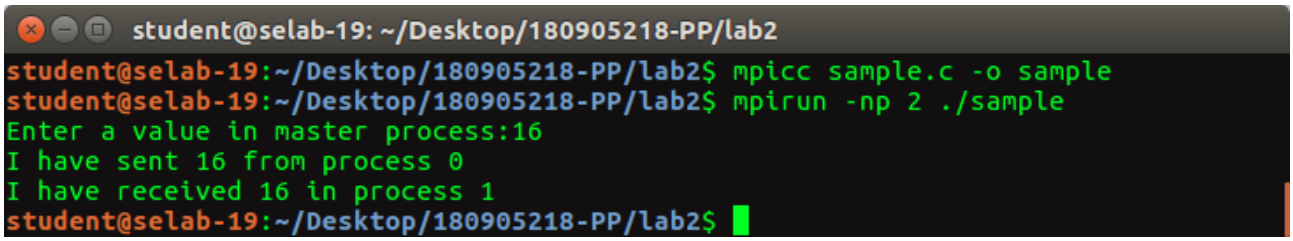
NAME: ARUNIMA SINGH THAKUR
SECTION: C
LAB NO: 2
LAB: PP
ROLL NO: 31
REGISTRATION NO: 180905218

SAMPLE PROGRAM:

Write a MPI program using standard send. The sender process sends a number to the receiver. The second process receives the number and prints it.

```
#include "mpi.h"
#include <stdio.h>
int main(int argc,char *argv[])
{
	int rank,size,x;
	MPI_Init(&argc,&argv);
	MPI_Comm_rank(MPI_COMM_WORLD,&rank);
	MPI_Comm_size(MPI_COMM_WORLD,&size);
	MPI_Status status;
	if(rank==0)
	{
		printf("Enter a value in master process:");
		scanf("%d",&x);
		MPI_Send(&x,1,MPI_INT,1,1,MPI_COMM_WORLD);
		fprintf(stdout,"I have sent %d from process 0\n",x);
		fflush(stdout);
	}
	else
	{
		MPI_Recv(&x,1,MPI_INT,0,1,MPI_COMM_WORLD,&status);
		fprintf(stdout,"I have received %d in process 1\n",x);
		fflush(stdout);
	}
	MPI_Finalize();
	return 0;
}
```



QUESTION 1:

Write a MPI program using synchronous send. The sender process sends a word to the receiver. The second process receives the word, toggles each letter of the word and sends it back to the first process. Both processes use synchronous send operations.

```
#include "mpi.h"
#include <stdio.h>
#include <string.h>
int main(int argc,char *argv[])
{
	int rank;
	char str[100];
	MPI_Init(&argc,&argv);
	MPI_Comm_rank(MPI_COMM_WORLD,&rank);
	MPI_Status status;

	if(rank==0)
	{
		printf("Enter a word in master process:");
		scanf("%s",str);
		MPI_Ssend(str,strlen(str),MPI_CHAR,1,0,MPI_COMM_WORLD);
		printf("I have sent %s from process 0\n",str);
		MPI_Recv(str,100,MPI_CHAR,1,0,MPI_COMM_WORLD,&status);
		printf("I have received %s in process 0\n",str);
	}
	else
	{
		MPI_Recv(str,100,MPI_CHAR,0,0,MPI_COMM_WORLD,&status);
		printf("I have received %s in process 1\n",str);
		for(int i=0;i<strlen(str);i++)
		{
			if(str[i]>='A' && str[i]<='Z') str[i] += 32;
			else if(str[i]>='a' && str[i]<='z') str[i] -=32;
		}
		MPI_Ssend(str,strlen(str),MPI_CHAR,0,0,MPI_COMM_WORLD);
		printf("I have sent %s from process 1\n",str);
	}
	MPI_Finalize();
	return 0;
}
```

```
student@selab-19: ~/Desktop/180905218-PP/lab2
student@selab-19:~/Desktop/180905218-PP/lab2$ mpicc ques1.c -o ques1
student@selab-19:~/Desktop/180905218-PP/lab2$ mpirun -np 2 ./ques1
Enter a word in master process: MaNIpaL
I have sent MaNIpaL from process 0
I have received mAniPAl in process 0
I have received MaNIpaL in process 1
I have sent mAniPAl from process 1
student@selab-19:~/Desktop/180905218-PP/lab2$
```

QUESTION 2:

Write a MPI program where the master program (process 0) sends a number to each of the slaves and the slave processes receive the number and prints it. Use standard send.

```
#include "mpi.h"
#include <stdio.h>
#include <string.h>
int main(int argc,char *argv[])
{
	int rank,size,x=100;
	MPI_Init(&argc,&argv);
	MPI_Comm_rank(MPI_COMM_WORLD, &rank);
	MPI_Comm_size(MPI_COMM_WORLD, &size);
	MPI_Status status;
	if(rank==0)
	{
		for(int i=1;i<size;i++)
		{
			MPI_Send(&x,1,MPI_INT,i,i,MPI_COMM_WORLD);
			printf("Process 0 has sent number to process %d. \n",i);
		}
	}
	else
	{
		MPI_Recv(&x,1,MPI_INT,0,rank,MPI_COMM_WORLD,&status);
		printf("Process %d has received number - %d. \n",rank,x);
	}
	MPI_Finalize();
	return 0;
}
```

```
student@selab-19: ~/Desktop/180905218-PP/lab2
student@selab-19:~/Desktop/180905218-PP/lab2$ mpicc ques2.c -o ques2
student@selab-19:~/Desktop/180905218-PP/lab2$ mpirun -np 6 ./ques2
Process 0 has sent number to process 1.
Process 0 has sent number to process 2.
Process 0 has sent number to process 3.
Process 0 has sent number to process 4.
Process 0 has sent number to process 5.
Process 2 has received number - 100.
Process 3 has received number - 100.
Process 5 has received number - 100.
Process 1 has received number - 100.
Process 4 has received number - 100.
student@selab-19:~/Desktop/180905218-PP/lab2$
```

QUESTION 3:

Write a MPI program to read N elements of the array in the root process (process 0) where N is equal to the total number of process. The root process sends one value to each of the slaves. Let even ranked process finds square of the received element and odd ranked process finds cube of received element. Use Buffered send.

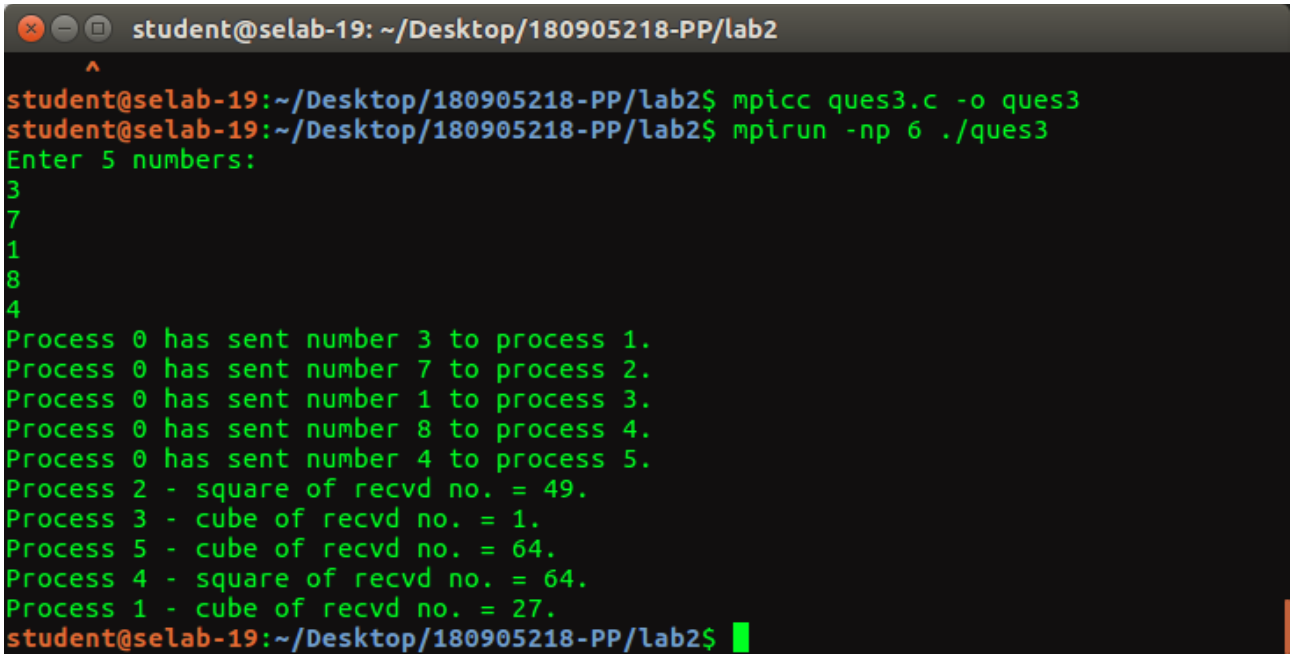
```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <mpi.h>

int main(int argc, char *argv[])
{
   int rank, size, x;
   MPI_Init(&argc, &argv);
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
   MPI_Comm_size(MPI_COMM_WORLD, &size);
   MPI_Status status;

   if(rank==0)
   {
       int arr[size-1];
       printf("Enter %d numbers:\n",size-1);
       for(int i=0;i<size-1;i++)
               scanf("%d",&arr[i]);

       int buff[100];
       int size_b=sizeof(buff);
       MPI_Buffer_attach(buff,size_b);
       for(int i=1;i<size;i++)
       {
```

```
            MPI_Bsend(&arr[i-1],1,MPI_INT,i,i,MPI_COMM_WORLD);
            printf("Process 0 has sent number %d to process %d.\n",arr[i-1],i);
        }
        MPI_Buffer_detach(buff,&size_b);
    }
    else if(rank%2==0)
    {
        MPI_Recv(&x,1,MPI_INT,0,rank,MPI_COMM_WORLD,&status);
        printf("Process %d - square of recvd no. = %d.\n",rank,x*x);
    }
    else
    {
        MPI_Recv(&x,1,MPI_INT,0,rank,MPI_COMM_WORLD,&status);
        printf("Process %d - cube of recvd no. = %d.\n",rank,x*x*x);
    }
    MPI_Finalize();
    return 0;
}
```



QUESTION 4:

Write a MPI program to read an integer value in the root process. Root process sends this value to Process1, Process1 sends this value to Process2 and so on. Last process sends the value back to root process. When sending the value each process will first increment the received value by one. Write the program using point to point communication routines.

```
#include <stdlib.h>
```

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

int main(int argc, char *argv[])
{
   int rank, size, x;
   MPI_Init(&argc, &argv);
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
   MPI_Comm_size(MPI_COMM_WORLD, &size);
   MPI_Status status;
   if (rank == 0)
   {
        printf("Enter a number: ");
        scanf("%d",&x);
        MPI_Send(&x,1,MPI_INT,1,rank+1,MPI_COMM_WORLD);
        printf("Process 0 has sent number %d to process 1.\n",x);
        MPI_Recv(&x,1,MPI_INT,size-1,0,MPI_COMM_WORLD,&status);
        printf("Process 0 received %d.\n",x);
        }
        else
        {
                MPI_Recv(&x,1,MPI_INT,rank-
1,rank,MPI_COMM_WORLD,&status);
                x++;
                printf("Process %d received %d, sending %d to process %d.\n",rank,x-
1,x,(rank+1)%size);
                MPI_Send(&x,1,MPI_INT,(rank+1)%size,
(rank+1)%size,MPI_COMM_WORLD);
        }
   MPI_Finalize();
   return 0;
}
```

```
student@selab-19: ~/Desktop/180905218-PP/lab2
student@selab-19:~/Desktop/180905218-PP/lab2$ mpicc ques4.c -o ques4
student@selab-19:~/Desktop/180905218-PP/lab2$ mpirun -np 6 ./ques4
Enter a number: 5
Process 0 has sent number 5 to process 1.
Process 1 received 5, sending 6 to process 2.
Process 2 received 6, sending 7 to process 3.
Process 3 received 7, sending 8 to process 4.
Process 4 received 8, sending 9 to process 5.
Process 5 received 9, sending 10 to process 0.
Process 0 received 10.
student@selab-19:~/Desktop/180905218-PP/lab2$
```