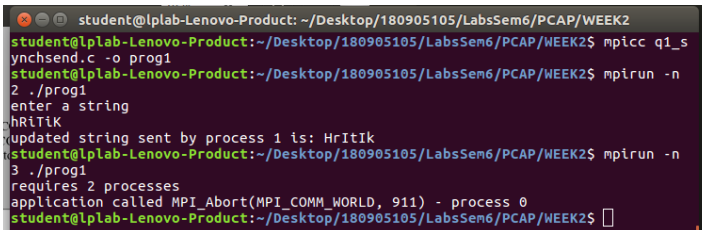
LAB 2:
HRITIK AJAY BANSAL
CSE A 15
180905105

Q1)
```
#include<mpi.h>
#include<stdio.h>
#include<string.h>
#define  comm MPI_COMM_WORLD
int main(int argc, char const *argv[])
{
	char str[10];
	MPI_Status status;
	int len;
	int rank,size;
	MPI_Init(NULL,NULL);
	MPI_Comm_rank(comm, &rank);
	MPI_Comm_size(comm,&size);
	if (size!=2)
	{
		printf("requires 2 processes \n");
		MPI_Abort(comm,911);
	}
	if (rank==0)
	{
		printf("enter a string \n");
		scanf("%s",str);
		len=strlen(str);

		MPI_Ssend(&len,1,MPI_INT,1,0,comm);
		MPI_Ssend(&str,len+1,MPI_CHAR,1,1,comm);

		MPI_Recv(&str,len+1,MPI_CHAR,1,3,comm,&status);
		printf("updated string sent by process 1 is: %s \n",str);

	}

	if (rank==1)
	{
		MPI_Recv(&len,1,MPI_INT,0,0,comm,&status);

		MPI_Recv(&str,len+1,MPI_CHAR,0,1,comm,&status);

		for (int i = 0; i < len; ++i)
		{
			if (str[i]>='A'&&str[i]<='Z')
				{
					str[i]+=32;

				}
				else
					str[i]-=32;
```

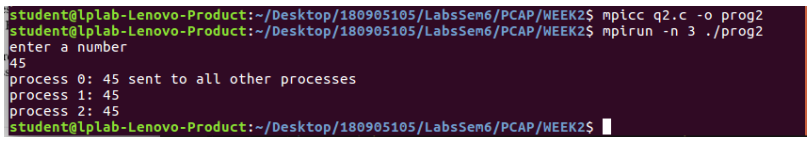
```
                }
                MPI_Ssend(&str,len+1,MPI_CHAR,0,3,comm);
        }
        MPI_Finalize();
        return 0;
}
```

OUTPUT:



Q2)
```
#include<mpi.h>
#include<stdio.h>
#include<string.h>
#define  comm MPI_COMM_WORLD
 int main(int argc, char const *argv[])
 {
        int root=0;
        int a;
        int rank,size;
        MPI_Init(NULL, NULL);
        MPI_Status status;
        MPI_Comm_rank(comm,&rank);
        MPI_Comm_size(comm,&size);

        if (rank==root)
        {
                printf("enter a number\n");
                scanf("%d",&a);
                 for (int i = 1; i < size; ++i)
                 {
                        MPI_Send(&a,1,MPI_INT,i,i,comm);
                 }
                 printf("process %d: %d sent to all other processes\n",rank,a);
        }
        else{
                MPI_Recv(&a,1,MPI_INT,0,rank,comm,&status);
                printf("process %d: %d\n",rank,a);
        }

        MPI_Finalize();
        return 0;
 }
```

output:



Q3)

```
#include<mpi.h>
#include<stdio.h>
#include<string.h>
#define  comm MPI_COMM_WORLD
 int main(int argc, char const *argv[])
 {
        int root=0;
        int a;
        int rank,size;
        MPI_Init(NULL, NULL);
        MPI_Status status;
        MPI_Comm_rank(comm,&rank);
        MPI_Comm_size(comm,&size);
        int b[size];
        char buff[100];
        int sl=100;
        MPI_Buffer_attach(buff,sl);
        if (rank==root)
        {
                printf("enter an arry of length %d \n",size);
                for (int i = 0; i < size; ++i)
                {
                        scanf("%d",&b[i]);
                }

                 for (int i = 1; i < size; ++i)
                 {
                        a=b[i];

                        MPI_Bsend(&a,1,MPI_INT,i,i,comm);

                 }
                 printf("process %d: Array elements sent to respective processes\n",rank);
        }
        else
                if (rank%2==0)
                {
                        MPI_Recv(&a,1,MPI_INT,0,rank,comm,&status);
                        printf("process %d: %d recieved\n",rank,a);
                        int x=a*a;
                        printf("process %d: square: %d \n",rank,x);
                }
                else
                {       MPI_Recv(&a,1,MPI_INT,0,rank,comm,&status);
                        printf("process %d: %d recieved\n",rank,a);
                        int x=a*a*a;
                        printf("process %d: cube: %d \n",rank,x);
        }
        MPI_Buffer_detach(&buff,&sl);
        MPI_Finalize();
        return 0;
 }
```

OUTPUT:

```
student@lplab-Lenovo-Product:~/Desktop/180905105/LabsSem6/PCAP/WEEK2$ mpicc q3_bufferedSe
nd.c -o prog3
student@lplab-Lenovo-Product:~/Desktop/180905105/LabsSem6/PCAP/WEEK2$ mpirun -n 5 ./prog3
enter an arry of length 5
0
1
2
3
4
process 0: Array elements sent to respective processes
process 2: 2 recieved
process 2: square: 4
process 3: 3 recieved
process 3: cube: 27
process 4: 4 recieved
process 4: square: 16
process 1: 1 recieved
process 1: cube: 1
```

Q4)

// program works the same for MPI_Send() and MPI_Ssend()

```
#include<mpi.h>
#include<stdio.h>
#include<string.h>
#define  comm MPI_COMM_WORLD
 int main(int argc, char const *argv[])
 {
        int root=0;
        int a;
        int rank,size;
        MPI_Init(NULL, NULL);
        MPI_Status status;
        MPI_Comm_rank(comm,&rank);
        MPI_Comm_size(comm,&size);

        if (rank==root)
        {
                printf("enter a number\n");
                scanf("%d",&a);

                MPI_Ssend(&a,1,MPI_INT,rank+1,rank,comm);
                printf("process %d: %d sent to process %d\n",rank,a,rank+1);
                MPI_Recv(&a,1,MPI_INT,size-1,size-1,comm,&status);
                printf("process %d: %d recieved from process %d\n",rank,a,size-1);
        }
        else{
                MPI_Recv(&a,1,MPI_INT,rank-1,rank-1,comm,&status);
                a=a+1;
                MPI_Ssend(&a,1,MPI_INT,(rank+1)%size,rank,comm);
                printf("process %d: %d sent to process %d\n",rank,a,(rank+1)%size);

        }

        MPI_Finalize();
        return 0;
 }
```

output:

```
student@lplab-Lenovo-Product:~/Desktop/180905105/LabsSem6/PCAP/WEEK2$ mpicc q4.c -o prog4
student@lplab-Lenovo-Product:~/Desktop/180905105/LabsSem6/PCAP/WEEK2$ mpirun -n 4 ./prog4
enter a number
23
process 0: 23 sent to process 1
process 0: 26 recieved from process 3
process 1: 24 sent to process 2
process 2: 25 sent to process 3
process 3: 26 sent to process 0
```