Parth Shukla
Lab 2
190905104

1)

```
#include "mpi.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
int main(int argc, char *argv[])
{
   int size, rank;
   MPI_Status status;
   MPI_Init(&argc, &argv);
   MPI_Comm_size(MPI_COMM_WORLD, &size);
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
   char word[5], y[5];
   int len = 5 * sizeof(char);
   if (rank == 0)
   {
      scanf("%s", word);
      MPI_Ssend(word, len, MPI_CHAR, 1, 101, MPI_COMM_WORLD);
      printf("Process %d sent: %s\n", rank, word);
      MPI_Recv(word, len, MPI_CHAR, 1, 102, MPI_COMM_WORLD,
            &status);
      printf("Process %d received: %s\n", rank, word);
   }
   else
   {
      MPI_Recv(y, len, MPI_CHAR, 0, 101, MPI_COMM_WORLD,
            &status);
      printf("Process %d received: %s\n", rank, y);
      for (int i = 0; i < strlen(y); i++)
      {
         if (y[i] >= 'A' && y[i] <= 'Z')
            y[i] += 32;
         else if (y[i] >= 'a' && y[i] <= 'z')
            y[i] -= 32;
      }
      sleep(1);
      MPI_Ssend(y, len, MPI_CHAR, 0, 102, MPI_COMM_WORLD);
      printf("Process %d sent: %s\n", rank, y);
   }
   MPI_Finalize();
}
```

```
student@dblab-hp-280-10:~/190905104_ParthShukla_PCAP/week2$ mpicc q1.c
student@dblab-hp-280-10:~/190905104_ParthShukla_PCAP/week2$ mpirun -n 2 ./a.out
hEllO
Process 0 sent: hEllO
Process 1 received: hEllO
Process 0 received: HeLLo
Process 1 sent: HeLLo
```

2)

```
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>
#define SIZE sizeof(int)

int main(int argc, char *argv[])
{
   int size, rank;
   MPI_Status status;
   MPI_Init(&argc, &argv);
   MPI_Comm_size(MPI_COMM_WORLD, &size);
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
   int *number = (int *)malloc(SIZE);
   int i;
   if (rank == 0)
   {
      *number = 5;
      for (i = 1; i < size; ++i)
      {
         printf("%d. Sent to %d: %d\n", rank, i, *number);
         // Send to the process with ID = i
         MPI_Send(number, SIZE, MPI_INT, i, 100 + i,
               MPI_COMM_WORLD);
      }
   }
   else
   {
      // Revc from the process with ID = 0
      MPI_Recv(number, SIZE, MPI_INT, 0, 100 + rank, MPI_COMM_WORLD,
            &status);
      printf("%d. Recv: %d\n", rank, *number);
   }
   MPI_Finalize();
}
```

```
student@dblab-hp-280-10:~/190905104_ParthShukla_PCAP/week2$ mpicc q2.c
student@dblab-hp-280-10:~/190905104_ParthShukla_PCAP/week2$ mpirun -n 6 ./a.out
0. Sent to 1: 5
0. Sent to 2: 5
0. Sent to 3: 5
0. Sent to 4: 5
0. Sent to 5: 5
1. Recv: 5
3. Recv: 5
2. Recv: 5
4. Recv: 5
5. Recv: 5
```

3)

```
#include "mpi/mpi.h"
#include <stdio.h>
#define comm MPI_COMM_WORLD

int main(){
   int rank,size;
   int arr[10],num;
   MPI_Status status;
   MPI_Init(NULL,NULL);
   MPI_Comm_rank(comm,&rank);
   MPI_Comm_size(comm,&size);
   int buff[100];
   int buffsize=100;

   if(rank==0){
      printf("enter an array of %d numbers",size-1);
      for(int i=1;i<size;i++){
         scanf("%d",&arr[i]);
      }
      MPI_Buffer_attach(buff,buffsize);
      for(int i=1;i<size;i++){
         MPI_Bsend(&arr[i],1,MPI_INT,i,i,comm);
      }
      MPI_Buffer_detach(&buff,&buffsize);
   }
   else{
      MPI_Recv(&num,1,MPI_INT,0,rank,comm,&status);
      if(rank%2==0){
         printf("in rank %d and number received is %d\neven rank - squared number is %d\n\
n",rank,num,num*num);


      }
      else{
         printf("in rank %d and number received is %d\nodd rank - cubed number is %d\n\
n",rank,num,num*num*num);

      }

   }
```

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

```
student@dblab-hp-280-10:~/190905104_ParthShukla_PCAP/week2$ mpirun -n 6 ./a.out
enter an array of 5 numbers5 4 3 2 1
in rank 2 and number received is 4
even rank - squared number is 16

in rank 3 and number received is 3
odd rank - cubed number is 27

in rank 4 and number received is 2
even rank - squared number is 4

in rank 1 and number received is 5
odd rank - cubed number is 125

in rank 5 and number received is 1
odd rank - cubed number is 1
```

4)

```
#include <stdio.h>
#include <stdlib.h>
#include <mpi.h>
#include <string.h>
int main(int argc, char *argv[])
{
    int rank, size, num;
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    if (rank == 0)
    {
        printf("\nEnter a number -- ");
        scanf("%d", &num);
        printf("\nProcess 0 -- ");
        printf("\nSending integer %d\n", num);
        MPI_Send(&num, 1, MPI_INT, 1, 0, MPI_COMM_WORLD);
        MPI_Recv(&num, 1, MPI_INT, size - 1, 0, MPI_COMM_WORLD,
MPI_STATUS_IGNORE);
        printf("\nProcess 0 -- \nReceived int %d\n", num);
    }
    else
    {
        MPI_Recv(&num, 1, MPI_INT, rank - 1, 0, MPI_COMM_WORLD,
MPI_STATUS_IGNORE);
        printf("\nProcess %d -- ", rank);
        printf("\nReceived integer %d", num);
        num++;
        printf("\nSending integer %d\n", num);
```

```
        MPI_Send(&num, 1, MPI_INT, (rank + 1) % size, 0, MPI_COMM_WORLD);
    }
    MPI_Finalize();
    return 0;
}
```