Q1.

#include<stdio.h>

#include<stdlib.h>

#include<mpi.h>

#include<string.h>

int main(int argc,char\* argv[]){

int rank,size;

char st[10];

char res[10];

MPI\_Init(&argc,&argv);

MPI\_Comm\_rank(MPI\_COMM\_WORLD,&rank);

MPI\_Comm\_size(MPI\_COMM\_WORLD,&size);

if(size!=2){

printf("\nUse 2 processes\n");

MPI\_Abort(MPI\_COMM\_WORLD,EXIT\_FAILURE);

}

if(rank==0){

printf("Enter string to toggle\n");

gets(st);

printf("\nProcess[%d]: Sending string %s\n",rank,st);

MPI\_Ssend(st,9,MPI\_CHAR,1,0,MPI\_COMM\_WORLD);

MPI\_Recv(res,9,MPI\_CHAR,1,0,MPI\_COMM\_WORLD,MPI\_STATUS\_IGNORE);

printf("\nProcess[%d]: Received string %s\n",rank,res);

}else{

MPI\_Recv(st,9,MPI\_CHAR,0,0,MPI\_COMM\_WORLD,MPI\_STATUS\_IGNORE);

printf("\nProcess[%d]: Received string %s\n",rank,st);

for(int i=0;i<strlen(st);i++){

if(st[i]>='a'&&st[i]<='z') st[i]-=32;

else if(st[i]>='A'&&st[i]<='Z') st[i]+=32;

}

printf("\nProcess[%d]: Sending string %s\n",rank,st);

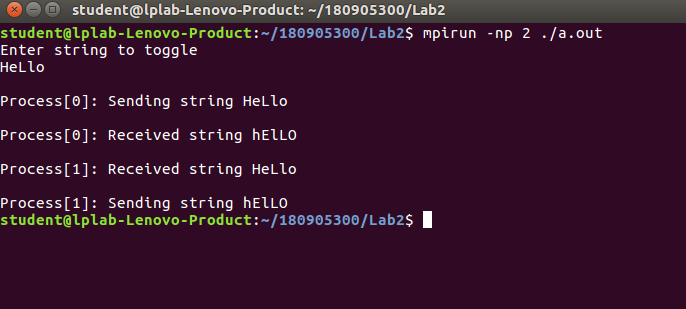
MPI\_Ssend(st,9,MPI\_CHAR,0,0,MPI\_COMM\_WORLD);

}

MPI\_Finalize();

return 0;

}



Q2.

#include<stdio.h>

#include<stdlib.h>

#include<mpi.h>

#include<string.h>

int main(int argc,char\* argv[]){

int rank,size,num;

char st[10];

char res[10];

MPI\_Init(&argc,&argv);

MPI\_Comm\_rank(MPI\_COMM\_WORLD,&rank);

MPI\_Comm\_size(MPI\_COMM\_WORLD,&size);

if(size!=4){

printf("\nUse 4 processes\n");

MPI\_Abort(MPI\_COMM\_WORLD,EXIT\_FAILURE);

}

if(rank==0){

printf("\nProcess[%d]: Sending numbers\n",rank);

num=1;

MPI\_Send(&num,1,MPI\_INT,1,0,MPI\_COMM\_WORLD);

num=2;

MPI\_Send(&num,1,MPI\_INT,2,0,MPI\_COMM\_WORLD);

num=3;

MPI\_Send(&num,1,MPI\_INT,3,0,MPI\_COMM\_WORLD);

}else{

MPI\_Recv(&num,1,MPI\_INT,0,0,MPI\_COMM\_WORLD,MPI\_STATUS\_IGNORE);

printf("\nProcess[%d]: Received %d\n",rank,num);

}

MPI\_Finalize();

return 0;

}



Q3.

#include<stdio.h>

#include<mpi.h>

#include<stdlib.h>

int main(int argc, char\* argv[])

{

int rank, size;

int k;

int buff\_size=MPI\_BSEND\_OVERHEAD+sizeof(int);

char\* buff=(char\*)malloc(buff\_size\*sizeof(int));

MPI\_Init(&argc, &argv);

MPI\_Comm\_rank(MPI\_COMM\_WORLD,&rank);

MPI\_Comm\_size(MPI\_COMM\_WORLD,&size);

int arr[7];

if(size!=8)

{

printf("\nUse 8 processes\n");

MPI\_Abort(MPI\_COMM\_WORLD,EXIT\_FAILURE);

}

if(rank==0)

{

MPI\_Buffer\_attach(buff,buff\_size);

printf("Please enter 7 integers.\n");

for(int i=0;i<7;i++)

{

scanf("%d",&arr[i]);

}

printf("Process[%d] sending the elements.\n",rank);

for(int i=1;i<8;i++)

{

MPI\_Bsend(&arr[i-1],1,MPI\_INT,i,0,MPI\_COMM\_WORLD);

}

}

else

{

MPI\_Recv(&k,1,MPI\_INT,0,0,MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

if(rank%2==0)

{

printf("Process[%d] recieved: %d and the square is: %d \n",rank, k,k\*k);

}

else

{

printf("Process[%d] recieved: %d and the cube is: %d \n",rank, k,k\*k\*k);

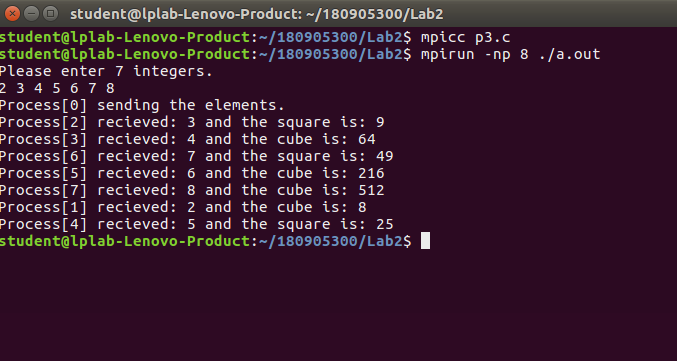
}

}

MPI\_Buffer\_detach(buff, &buff\_size);

MPI\_Finalize();

}



Q4.

#include<stdio.h>

#include<stdlib.h>

#include<mpi.h>

int main(int argc, char\* argv[])

{

int rank,size;

int k;

MPI\_Init(&argc, &argv);

MPI\_Comm\_rank(MPI\_COMM\_WORLD,&rank);

MPI\_Comm\_size(MPI\_COMM\_WORLD,&size);

if(size!=5)

{

printf("\nUse 5 processes\n");

MPI\_Abort(MPI\_COMM\_WORLD,EXIT\_FAILURE);

}

if(rank==0)

{

printf("Please enter the integer you want to send.\n");

scanf("%d",&k);

MPI\_Send(&k,1,MPI\_INT,1,0,MPI\_COMM\_WORLD);

}

else if(rank!=4)

{

MPI\_Recv(&k,1,MPI\_INT,rank-1,0,MPI\_COMM\_WORLD,MPI\_STATUS\_IGNORE);

printf("Process[%d] recieved: %d From %d\n", rank, k, rank-1);

k+=1;

MPI\_Send(&k,1,MPI\_INT,rank+1,0,MPI\_COMM\_WORLD);

}

else

{

MPI\_Recv(&k,1,MPI\_INT,rank-1,0,MPI\_COMM\_WORLD,MPI\_STATUS\_IGNORE);

printf("Process[%d] recieved: %d From %d\n", rank, k, rank-1);

k+=1;

MPI\_Send(&k,1,MPI\_INT,0,0,MPI\_COMM\_WORLD);

}

MPI\_Finalize();

}

