Lab 4

Dhruvik Chevli

Q1.

#include<stdio.h>

#include<stdlib.h>

#include<mpi.h>

long fact(int a){

if(a==0) return 1;

return a\*fact(a-1);

}

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

int rank,size,\*a=NULL,c;

long res,\*b=NULL;

MPI\_Init(&argc,&argv);

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

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

a=(int\*)malloc(size\*sizeof(int));

b=(long\*)malloc(size\*sizeof(long));

if(rank==0){

printf("Enter %d numbers.\n",size);

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

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

}

MPI\_Scatter(a,1,MPI\_INT,&c,1,MPI\_INT,0,MPI\_COMM\_WORLD);

res=fact(c);

printf("Process[%d]: Factorial of %d is %ld\n",rank,c,res);

MPI\_Gather(&res,1,MPI\_LONG,b,1,MPI\_LONG,0,MPI\_COMM\_WORLD);

if(rank==0){

res=0;

for(int i=0;i<size;i++) res+=b[i];

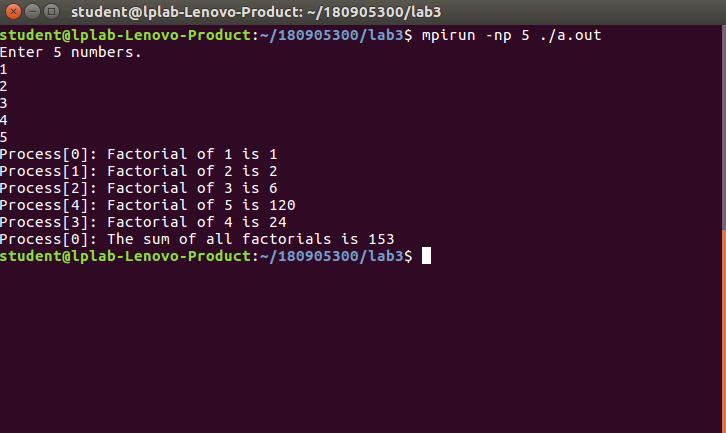
printf("Process[%d]: The sum of all factorials is %ld\n",rank,res);

}

MPI\_Finalize();

return 0;

}



Q2.

#include<stdio.h>

#include<stdlib.h>

#include<mpi.h>

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

int rank,size,\*a=NULL,\*c=NULL;

float res,\*b=NULL;

if(argc<1){

printf("Enter m as command line argument");

exit(EXIT\_FAILURE);

}

MPI\_Init(&argc,&argv);

int m=atoi(argv[1]);

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

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

a=(int\*)malloc(m\*size\*sizeof(int));

b=(float\*)malloc(size\*sizeof(float));

c=(int\*)malloc(m\*sizeof(int));

if(rank==0){

printf("Enter %d numbers.\n",m\*size);

for(int i=0;i<m\*size;i++)

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

}

MPI\_Scatter(a,m,MPI\_INT,c,m,MPI\_INT,0,MPI\_COMM\_WORLD);

res=0;

for(int i=0;i<m;i++){

res+=c[i];

}

res/=m;

printf("Process[%d]: Average is %f\n",rank,res);

MPI\_Gather(&res,1,MPI\_FLOAT,b,1,MPI\_FLOAT,0,MPI\_COMM\_WORLD);

if(rank==0){

res=0;

for(int i=0;i<size;i++) res+=b[i];

res/=size;

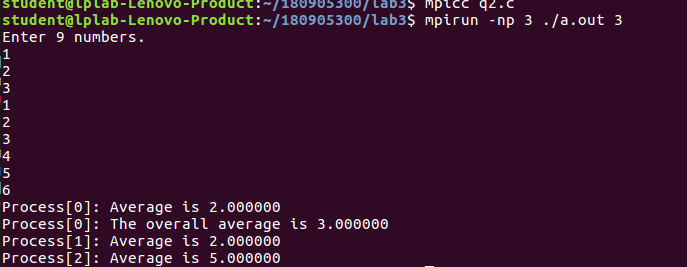
printf("Process[%d]: The overall average is %f\n",rank,res);

}

MPI\_Finalize();

return 0;

}



Q3.

#include<stdio.h>

#include<stdlib.h>

#include<mpi.h>

#include<string.h>

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

{

int rank,size;

int res;

int s;

MPI\_Init(&argc,&argv);

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

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

char st[100];

char\* c=(char\*)malloc(sizeof(char)\*100);

int\* b=(int\*)malloc(size\*sizeof(int));

if(rank==0)

{

scanf("%s",st);

int k=strlen(st);

if(k%size!=0)

{

printf("String length should be divisible by the number of processes.");

return 0;

}

s=k/size;

}

MPI\_Bcast(&s,1,MPI\_INT,0,MPI\_COMM\_WORLD);

MPI\_Scatter(st,s,MPI\_CHAR,c,s,MPI\_CHAR,0,MPI\_COMM\_WORLD);

res=0;

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

{

// printf("%c",c[i]);

if(c[i]=='a'||c[i]=='e'||c[i]=='i'||c[i]=='o'||c[i]=='u')

continue;

res+=1;

}

// printf("\nProcess[%d]: Non vowel count is %d\n",rank,res);

MPI\_Gather(&res,1,MPI\_INT,b,1,MPI\_INT,0,MPI\_COMM\_WORLD);

if(rank==0)

{

res=0;

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

{

res+=b[i];

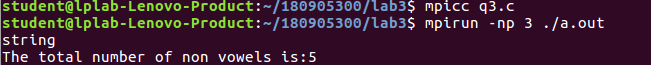
}

printf("The total number of non vowels is:%d\n",res);

}

MPI\_Finalize();

}



Q4.

#include<stdio.h>

#include<stdlib.h>

#include<mpi.h>

#include<string.h>

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

{

int rank,size;

int res;

int s;

MPI\_Init(&argc,&argv);

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

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

char st[100];

char st1[100];

char\* a=(char\*)malloc(sizeof(char)\*100);

char\* b=(char\*)malloc(100\*sizeof(char));

char\* c=(char\*)malloc(199\*sizeof(char));

char\* fst=(char\*)malloc(sizeof(char)\*200);

if(rank==0)

{

scanf("%s",st);

int k=strlen(st);

if(k%size!=0)

{

printf("String length should be divisible by the number of processes.");

return 0;

}

s=k/size;

scanf("%s",st1);

}

MPI\_Bcast(&s,1,MPI\_INT,0,MPI\_COMM\_WORLD);

MPI\_Scatter(st,s,MPI\_CHAR,a,s,MPI\_CHAR,0,MPI\_COMM\_WORLD);

res=0;

MPI\_Scatter(st1,s,MPI\_CHAR,b,s,MPI\_CHAR,0,MPI\_COMM\_WORLD);

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

{

c[i\*2]=a[i];

c[i\*2+1]=b[i];

}

MPI\_Gather(c,2\*s,MPI\_CHAR,fst,2\*s,MPI\_CHAR,0,MPI\_COMM\_WORLD);

if(rank==0)

{

printf("%s\n",fst);

}

MPI\_Finalize();

}

