**HRITIK BANSAL**

**CSE A 15**

**180905105**

**PP LAB**

**WEEK 4**

**Q1)**

#include "mpi.h"

#include <stdio.h> #include <string.h>

int fact(int n)

{

if (n <= 1) return 1;

else return n \* fact(n - 1);

}

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

{

int rank, size; int i = 0;

int k = 0, fac = 1, ans[1000], sum = 0; int n;

MPI\_Init(&argc, &argv);

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

// Set the error handler to MPI\_ERRORS\_RETURN

MPI\_Errhandler\_set(MPI\_COMM\_WORLD, MPI\_ERRORS\_RETURN);

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

//Get the error code on broadcasting; purposely fail this int error;

error = MPI\_Bcast(&fac, 1, MPI\_INT, 4, MPI\_COMM\_WORLD);

if (error != MPI\_SUCCESS)

{

char s[100]; int len, class1;

MPI\_Error\_string(error, s, &len); MPI\_Error\_class(error, &class1); fprintf(stderr, "Error description is %s", s); fflush(stderr);

fprintf(stderr, "Error class is %d", class1); fflush(stderr);

}

for (i = 1; i <= rank + 1; i++)

{

fac = fac \* i;

}

MPI\_Scan(&fac, &k, 1, MPI\_INT, MPI\_SUM, MPI\_COMM\_WORLD);

if (rank == size - 1)

{

fprintf(stdout, "%d\n", k); fflush(stdout);

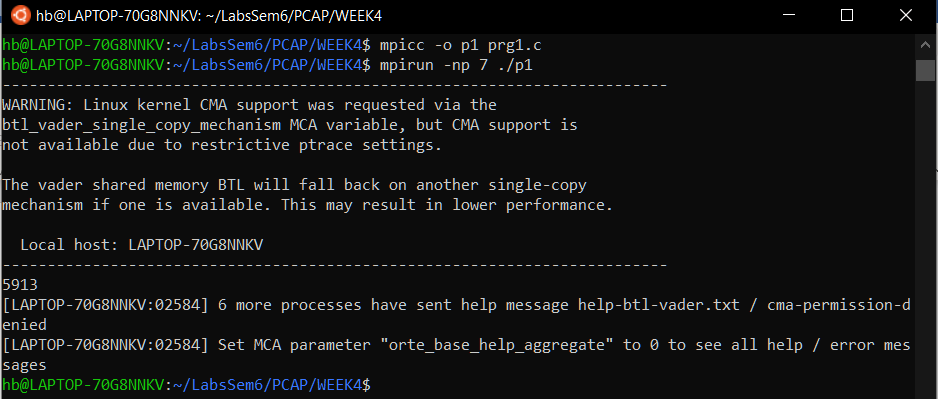
}

MPI\_Finalize();

return 0;

}

**Output:**

****

**Q2)**

#include "mpi.h"

#include <stdio.h>

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

{

int ierr,errclass,resultlen;

char err\_buffer[MPI\_MAX\_ERROR\_STRING];

int rank,size,i;

float rect ,pi;

MPI\_Init(&argc,&argv);

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

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

MPI\_Errhandler\_set(MPI\_COMM\_WORLD,MPI\_ERRORS\_RETURN);

float x = (rank+0.5)/size; float x2 = x\*x;

rect = (4/(1+x2))\*(1/(float)size);

ierr =MPI\_Reduce(&rect,&pi,1,MPI\_FLOAT,MPI\_SUM,0,MPI\_COMM\_WORLD );

if (ierr != MPI\_SUCCESS) {

MPI\_Error\_class(ierr,&errclass);

if (errclass== MPI\_ERR\_RANK) {

fprintf(stderr,"Invalid rank used in MPI send call\n"); MPI\_Error\_string(ierr,err\_buffer,&resultlen); printf("%s\n",err\_buffer);

MPI\_Abort(MPI\_COMM\_WORLD,ierr);

}

}

if(rank==0){

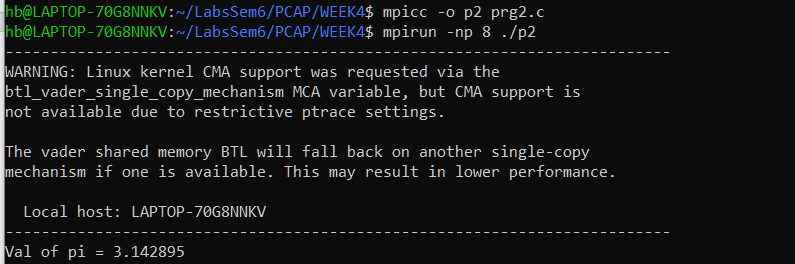
printf("Val of pi = %f\n",pi);

}

MPI\_Finalize();

return 0; }

**Output:**

****

**Q3)** #include "mpi.h" #include <stdio.h>

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

{

int ierr,errclass,resultlen;

char err\_buffer[MPI\_MAX\_ERROR\_STRING];

int rank,size,i,j,a[3][3],b[3],key,count,countsum; MPI\_Init(&argc,&argv);

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

if(rank==0){ printf(" Enter values in 3x3 matrix:\n"); for(int i=0;i<3;++i){ for(int j=0;j<3;++j){ scanf("%d",&a[i][j]);

}

}

printf(" Enter ele to search:\n");

scanf("%d",&key);

}

ierr = MPI\_Bcast(&key, 1, MPI\_INT, 0, MPI\_COMM\_WORLD);

if (ierr != MPI\_SUCCESS) { MPI\_Error\_class(ierr,&errclass); if (errclass== MPI\_ERR\_RANK) {

fprintf(stderr,"Invalid rank used in MPI send call\n"); MPI\_Error\_string(ierr,err\_buffer,&resultlen); printf("%s\n",err\_buffer);

MPI\_Abort(MPI\_COMM\_WORLD,ierr);

}

}

ierr=MPI\_Scatter(a,3,MPI\_INT,b,3,MPI\_INT,0,MPI\_COMM\_WORLD);

if (ierr != MPI\_SUCCESS) { MPI\_Error\_class(ierr,&errclass);

if (errclass== MPI\_ERR\_RANK) {

fprintf(stderr,"Invalid rank used in MPI send call\n"); MPI\_Error\_string(ierr,err\_buffer,&resultlen);

printf("%s\n",err\_buffer);

MPI\_Abort(MPI\_COMM\_WORLD,ierr);

}

}

for(int i=0;i<3;++i){ if(key==b[i]){

++ count;

}

}

ierr=MPI\_Reduce(&count,&countsum,1,MPI\_INT,MPI\_SUM,0,MPI\_COMM\_WORLD );

if (ierr != MPI\_SUCCESS) {

MPI\_Error\_class(ierr,&errclass);

if (errclass== MPI\_ERR\_RANK) {

fprintf(stderr,"Invalid rank used in MPI send call\n"); MPI\_Error\_string(ierr,err\_buffer,&resultlen);

printf("%s\n",err\_buffer);

MPI\_Abort(MPI\_COMM\_WORLD,ierr);

}

}

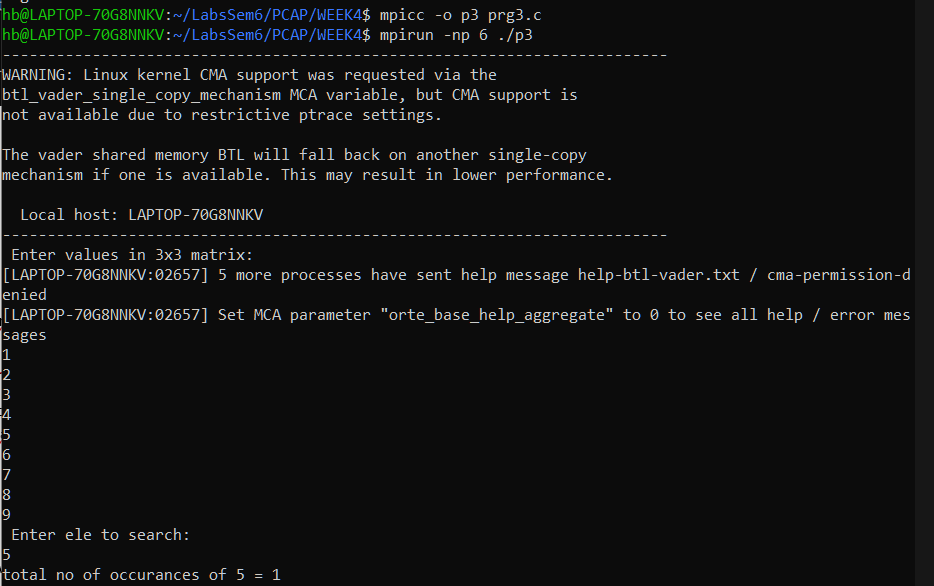
if(rank==0){

printf("total no of occurances of %d = %d\n",key,countsum);

}

MPI\_Finalize(); return 0; **}**

**Output:**

****

**Q4)** #include "mpi.h"

#include <stdio.h>

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

int ierr,errclass,resultlen;

char err\_buffer[MPI\_MAX\_ERROR\_STRING];

int rank,size,i,j,a[4][4],b[4],c[4],count=0,countsum;

MPI\_Init(&argc,&argv);

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

if(rank==0){

printf(" Enter values in 4x4 matrix:\n");

for(int i=0;i<4;++i){ for(int j=0;j<4;++j){ scanf("%d",&a[i][j]);

}

} }

ierr = MPI\_Scatter(a,4,MPI\_INT,b,4,MPI\_INT,0,MPI\_COMM\_WORLD);

if (ierr != MPI\_SUCCESS) {

MPI\_Error\_class(ierr,&errclass);

if (errclass== MPI\_ERR\_RANK) {

fprintf(stderr,"Invalid rank used in MPI send call\n"); MPI\_Error\_string(ierr,err\_buffer,&resultlen);

printf("%s\n",err\_buffer);

MPI\_Abort(MPI\_COMM\_WORLD,ierr);

}

}

ierr=MPI\_Scan(b,c,4,MPI\_INT,MPI\_SUM,MPI\_COMM\_WORLD );

if (ierr != MPI\_SUCCESS) { MPI\_Error\_class(ierr,&errclass);

if (errclass== MPI\_ERR\_RANK) {

fprintf(stderr,"Invalid rank used in MPI send call\n");

MPI\_Error\_string(ierr,err\_buffer,&resultlen);

printf("%s\n",err\_buffer);

MPI\_Abort(MPI\_COMM\_WORLD,ierr);

}

}

if(rank==0){

printf(" output 4x4 matrix:\n");

}

printf(" process %d :",rank); for(int i=0;i<4;++i){

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

}

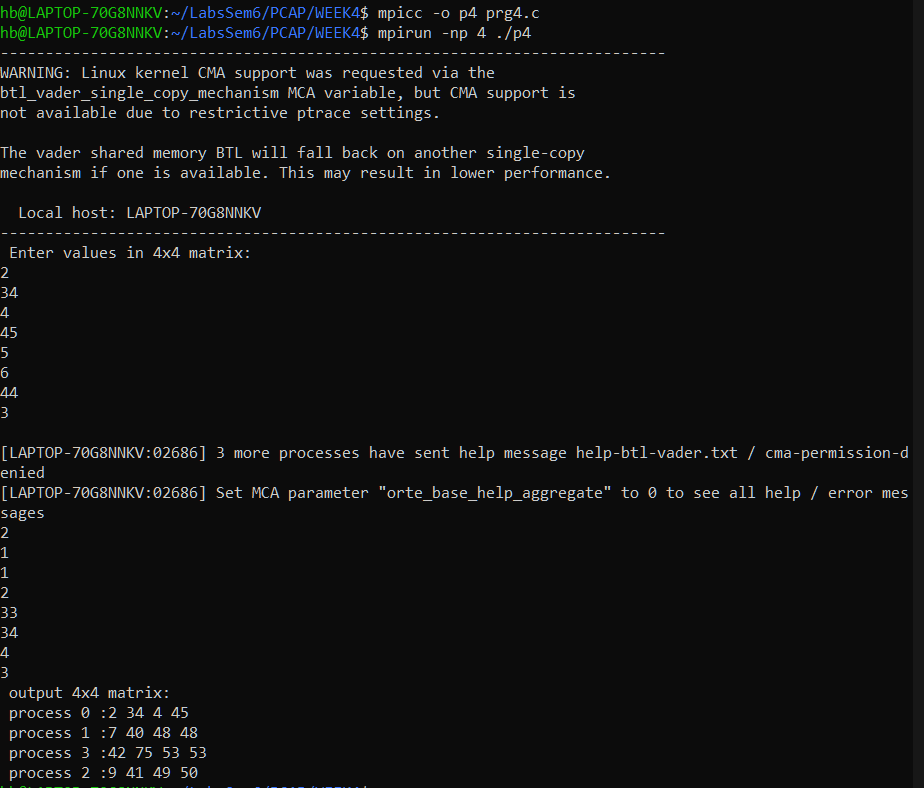
printf("\n");

MPI\_Finalize();

return 0;

}

**Output:**

****