**ARUNIMA SINGH THAKUR**

**31 CSE-C**

**180905218**

**PPLAB 4**

**2/5/21**

1. **Sum of factorials**

#include "mpi.h"

#include <stdio.h>

#include <string.h>

void ErrorHandler(int error\_code)

{

if(error\_code!=MPI\_SUCCESS)

{

char error\_string[BUFSIZ];

int reslen, error\_class;

MPI\_Error\_class(error\_code, &error\_class);

MPI\_Error\_string(error\_code, error\_string, &reslen);

printf("%d %s\n", error\_code, error\_string);

MPI\_Error\_string(error\_class, error\_string, &reslen);

printf("%d %s\n", error\_class, error\_string);

}

}

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

{

int rank, size;

MPI\_Init(&argc, &argv);

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

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

int c = 3;

int error\_code = MPI\_Comm\_size(c, &size);

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

ErrorHandler(error\_code);

MPI\_Status status;

int fact = 1;

int factsum = 0;

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

fact\*=i;

MPI\_Scan(&fact, &factsum, 1, MPI\_INT, MPI\_SUM,

MPI\_COMM\_WORLD);

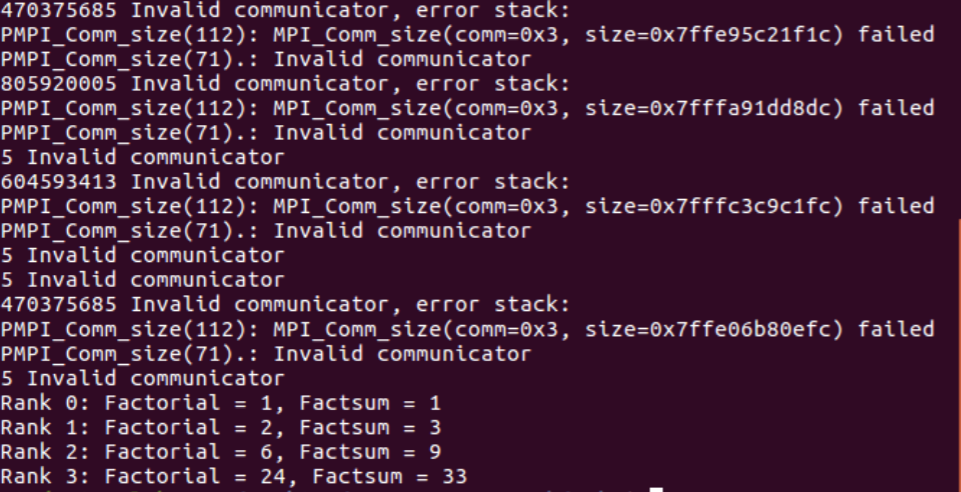
printf("Rank %d: Factorial = %d, Factsum = %d\n", rank, fact, factsum);

MPI\_Finalize();

return 0;

}

**OUTPUT:**



1. **Calculate PI**

#include "mpi.h"

#include <stdio.h>

#include <string.h>

void ErrorHandler(int error\_code)

{

if(error\_code!=MPI\_SUCCESS)

{

char error\_string[BUFSIZ];

int reslen, error\_class;

MPI\_Error\_class(error\_code, &error\_class);

MPI\_Error\_string(error\_code, error\_string, &reslen);

printf("%d %s\n", error\_code, error\_string);

MPI\_Error\_string(error\_class, error\_string, &reslen);

printf("%d %s\n", error\_class, error\_string);

}

}

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

{

int rank, size;

MPI\_Init(&argc, &argv);

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

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

int c = 3;

int error\_code = MPI\_Comm\_size(c, &size);

ErrorHandler(error\_code);

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

MPI\_Status status;

float val =0, pi=0;

val = (4.0/(1+((rank+0.5)/size)\*((rank+0.5)/size)))\*(1.0/size);

MPI\_Reduce(&val, &pi, 1, MPI\_FLOAT, MPI\_SUM,

0,MPI\_COMM\_WORLD);

if(rank==0)

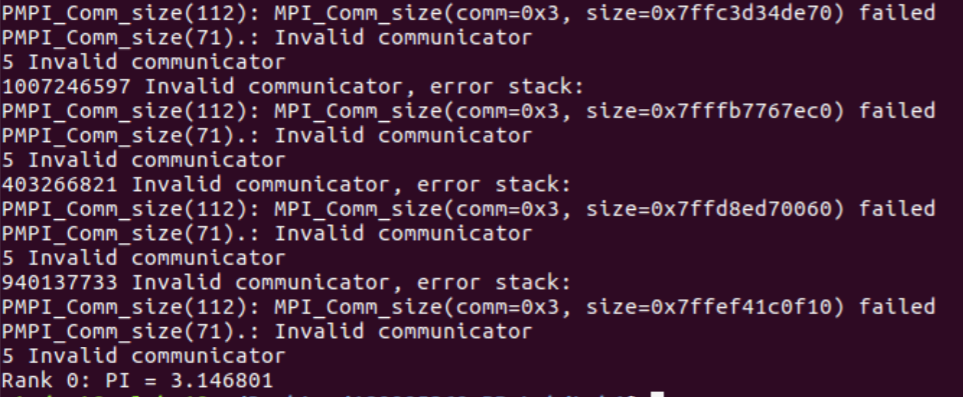
printf("Rank %d: PI = %f\n", rank, pi);

MPI\_Finalize();

return 0;

}

**OUTPUT:**



1. **Find value in 3x3 matrix**

#include "mpi.h"

#include <stdio.h>

#include <string.h>

void ErrorHandler(int error\_code)

{

if(error\_code!=MPI\_SUCCESS)

{

char error\_string[BUFSIZ];

int reslen, error\_class;

MPI\_Error\_class(error\_code, &error\_class);

MPI\_Error\_string(error\_code, error\_string, &reslen);

printf("%d %s\n", error\_code, error\_string);

MPI\_Error\_string(error\_class, error\_string, &reslen);

printf("%d %s\n", error\_class, error\_string);

}

}

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

{

int rank, size;

MPI\_Init(&argc, &argv);

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

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

int c = 3;

int error\_code = MPI\_Comm\_size(c, &size);

ErrorHandler(error\_code);

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

MPI\_Status status;

int count = 0;

int total\_count = 0;

int matrix[3][3];

int recvbuf[3];

int val;

if(rank==0)

{

printf("Enter 3x3 values below:\n");

for(int i = 0; i<3; i++)

for(int j=0; j<3; j++)

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

printf("\nEnter value to search for below:\n");

scanf(" %d", &val);

}

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

MPI\_Scatter(matrix, 3, MPI\_INT, recvbuf, 3, MPI\_INT, 0,

MPI\_COMM\_WORLD);

for(int i = 0; i<3; i++)

if(recvbuf[i]==val)

count++;

MPI\_Reduce(&count, &total\_count, 1, MPI\_INT, MPI\_SUM,

0,MPI\_COMM\_WORLD);

if(rank==0)

printf("Rank %d: Total count of %d in the matrix = %d\n", rank, val,

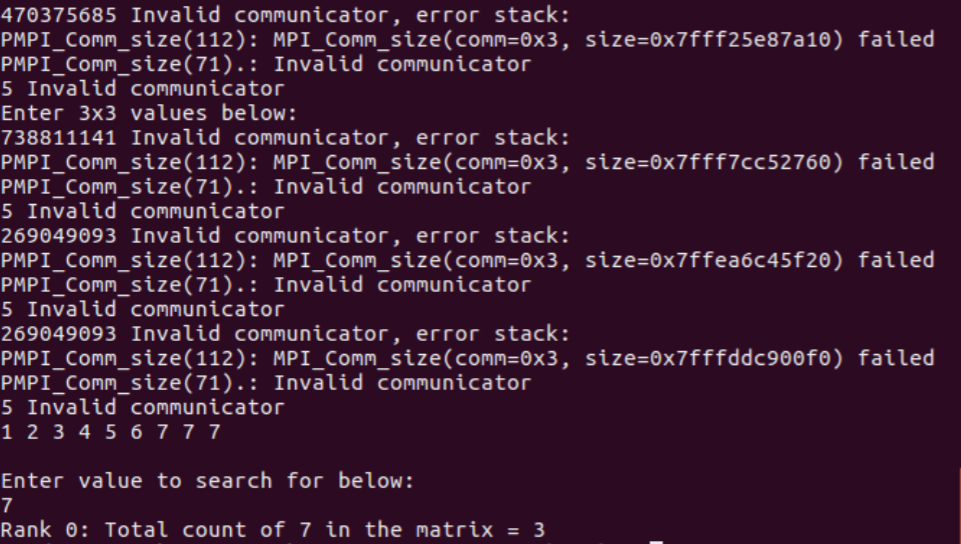
total\_count);

MPI\_Finalize();

return 0;

}

**OUTPUT:**



1. **Prev row sum 4x4 matrix**

#include "mpi.h"

#include <stdio.h>

#include <string.h>

void ErrorHandler(int error\_code)

{

if(error\_code!=MPI\_SUCCESS)

{

char error\_string[BUFSIZ];

int reslen, error\_class;

MPI\_Error\_class(error\_code, &error\_class);

MPI\_Error\_string(error\_code, error\_string, &reslen);

printf("%d %s\n", error\_code, error\_string);

MPI\_Error\_string(error\_class, error\_string, &reslen);

printf("%d %s\n", error\_class, error\_string);

}

}

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

{

int rank, size;

MPI\_Init(&argc, &argv);

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

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

int c = 3;

int error\_code = MPI\_Comm\_size(c, &size);

ErrorHandler(error\_code);

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

MPI\_Status status;

int count = 0;

int total\_count = 0;

int matrix[4][4];

int recvbuf[4];

int val;

int sum[4];

if(rank==0)

{

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

for(int i = 0; i<4; i++)

for(int j=0; j<4; j++)

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

}

MPI\_Scatter(matrix, 4, MPI\_INT, recvbuf, 4, MPI\_INT, 0,

MPI\_COMM\_WORLD);

MPI\_Scan(recvbuf, sum, 4, MPI\_INT, MPI\_SUM,MPI\_COMM\_WORLD);

for(int i =0; i<4; i++)

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

printf("\n");

MPI\_Finalize();

return 0;

}

**OUTPUT:**

