**PP Lab 4: Collective Communication and Error Handling in MPI**

**Name:** GP Anirudh

**Roll Number:** 59

**Section:** B

**Batch:** B2

**Registration Number:** 180905452

**Date:** 16-3-2021

1.) Write a MPI program using N processes to find 1! + 2! +.....+N! Use scan. Also, handle different errors using error handling routines.

**Code:**

#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,&ans[i],1,MPI\_INT,MPI\_SUM,MPI\_COMM\_WORLD);

}

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

if (rank == size-1)

{

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

fflush(stdout);

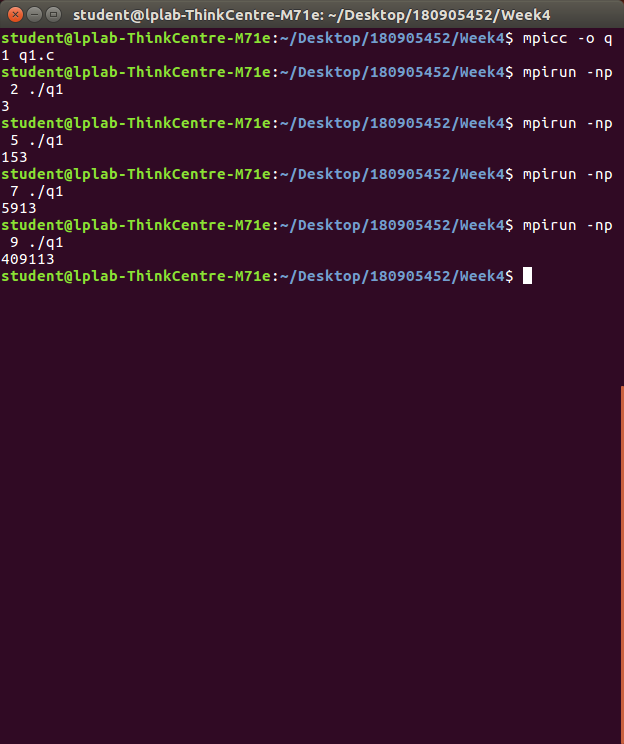
}

MPI\_Finalize();

return 0;

}

**Output:**



2.) Write a MPI program to calculate π-value by integrating f(x) = 4 /(1+x2). Area under the curve is divided into rectangles and the rectangles are distributed to the processors. Also handle different errors using error handling routines.

**Code:**

#include <mpi.h>

#include <stdio.h>

#include <string.h>

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

{

int rank, size;

int i = 0, j;

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

int n, a[100][100], b[100];

float x, y, area, pi1;

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);

int error = MPI\_Bcast(&size, 1, MPI\_INT, 0, 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);

}

x = (float)(rank+1)/size;

y = 4.f/(1+x\*x);

area = (1/(float)size)\*y;

MPI\_Reduce(&area, &pi1, 1, MPI\_FLOAT, MPI\_SUM, 0, MPI\_COMM\_WORLD);

if (rank == 0)

{

fprintf(stdout, "%f\n", pi1);

fflush(stdout);

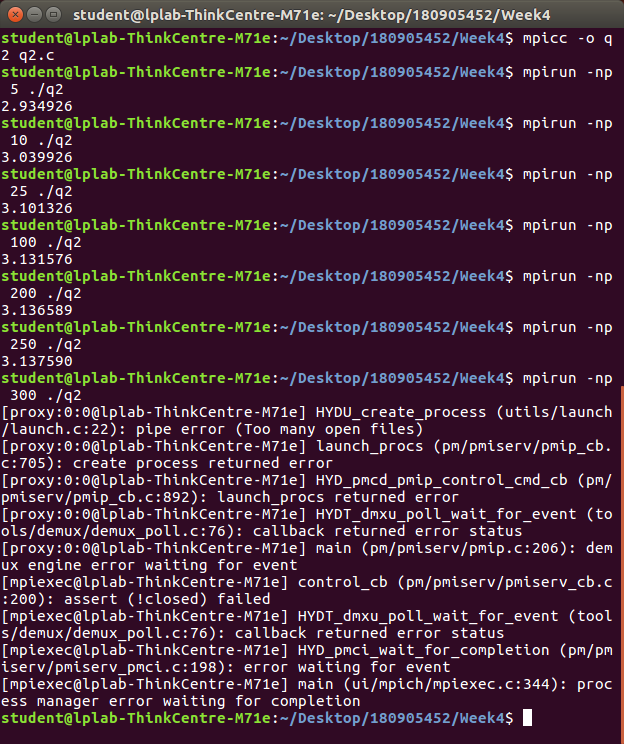
}

MPI\_Finalize();

return 0;

}

**Output:**



3.) Write a MPI program to read a 3 X 3 matrix. Enter an element to be searched in the root process. Find the number of occurrences of this element in the matrix using three processes.

**Code:**

#include <stdio.h>

#include <mpi.h>

void ErrorHandler(int error\_code)

{

char error\_string[MPI\_MAX\_ERROR\_STRING];

int length\_of\_error\_string,error\_class;

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

MPI\_Error\_string(error\_code,error\_string,&length\_of\_error\_string);

if(error\_code!=0)

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

}

void main (int a,char \*b[])

{

int rank, ele,size;

int a1[3][3];

int b1[3];

int count=0;

int tc = 0;

int error\_code;

MPI\_Init(&a, &b);

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

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

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

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

ErrorHandler(error\_code);

if (rank == 0)

{

printf("Enter elements into matrix\n");

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

{

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

{

scanf("%d", &a1[i][j]);

}

}

printf("Enter element to count \n");

scanf("%d", &ele);

}

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

MPI\_Scatter(a1 , 3, MPI\_INT, b1, 3, MPI\_INT, 0, MPI\_COMM\_WORLD);

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

{

printf("%d ",b1[i]);

if (b1[i] == ele)

{

count ++;

}

}

printf("\nProcess %d found %d occurrences\n",rank,count);

MPI\_Reduce(&count, &tc, 1, MPI\_INT, MPI\_SUM, 0, MPI\_COMM\_WORLD);

if (rank == 0)

{

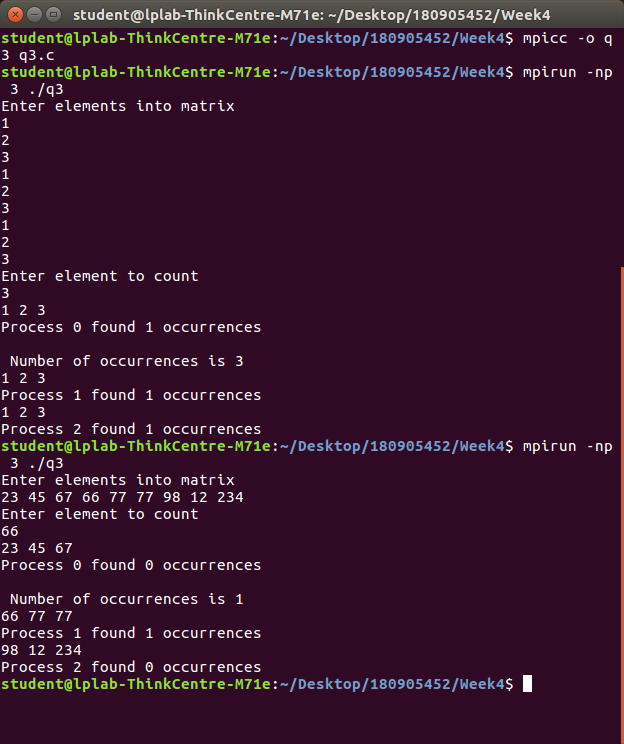
printf("\nNumber of occurrences is %d\n",tc);

}

MPI\_Finalize();

}

**Output:**



4.) Write a MPI program to read 4 X 4 matrix and display the following output using four processes.

**Code:**

#include <mpi.h>

#include <stdio.h>

#include <string.h>

void ErrorHandler(int error\_code)

{

if (error\_code != MPI\_SUCCESS)

{

char error\_string[BUFSIZ];

int length\_of\_error\_string, error\_class;

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

MPI\_Error\_string(error\_code, error\_string, &length\_of\_error\_string);

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

}

}

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

{

int rank, size, error\_code;

int i = 0, j;

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

int n, a[100][100], b[100];

MPI\_Init(&argc, &argv);

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

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

if (rank == 0)

{

printf("Enter the elements of i/p matrix \n");

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

{

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

{

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

}

}

printf("\n");

}

error\_code = MPI\_Scatter(a, 100, MPI\_INT, b, 100, MPI\_INT, 0, MPI\_COMM\_WORLD);

error\_code = MPI\_Scan(b, ans, 4, MPI\_INT, MPI\_SUM, MPI\_COMM\_WORLD);

ErrorHandler(error\_code);

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

{

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

}

printf("\n");

MPI\_Finalize();

return 0;

}

**Output:**

