PP Lab 4: Collective Communication and Error Handling in MPI

Name: Rajvardhan Reddy Nandyala
Roll Number: 19
Section: B
Batch: B1
Registration Number: 180905093

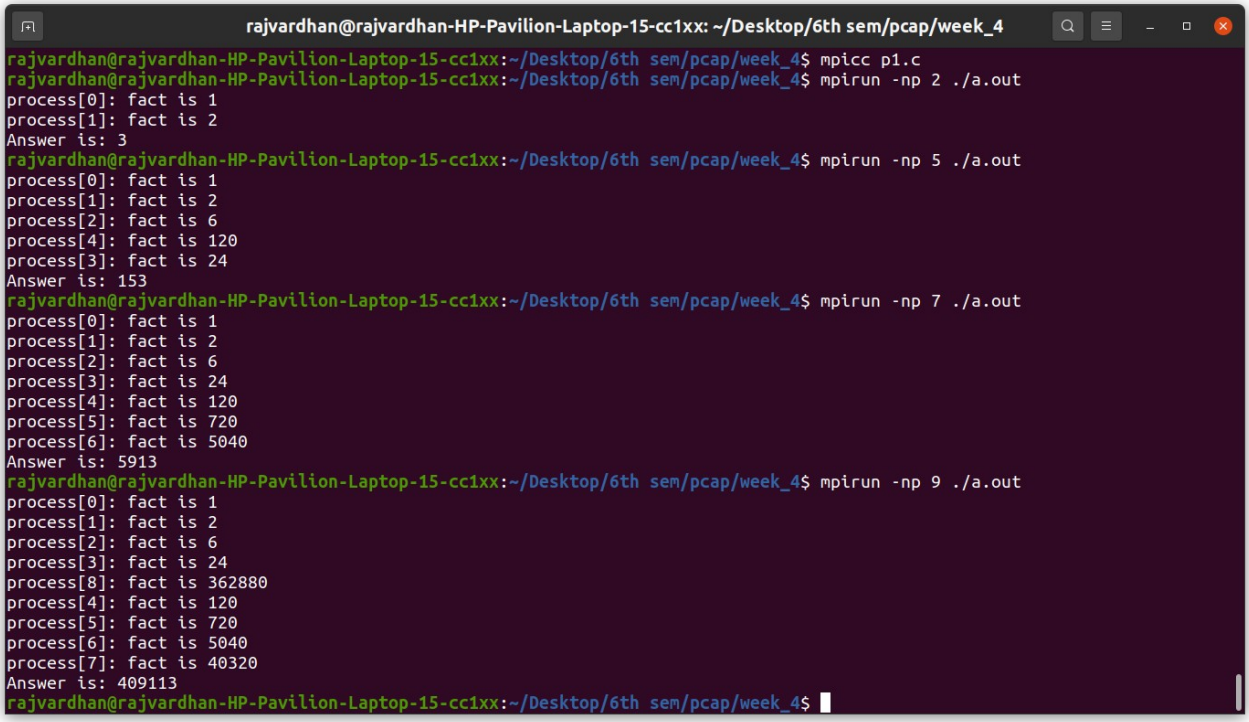
1.) Write a MPI program using N processes to find 1! + 2! +.....+N! Use scan. Also, handle different errors using error handling routines.

Code:

```
#include <mpi.h>
#include <stdio.h>
#include <string.h>
#define BUFSIZE 100

void Error_Handler(int error_code);

int main(int argc, char *argv[])
{
	int size, rank;

	int sum=0;
	int fact=1;
	MPI_Init(&argc, &argv);
	MPI_Errhandler_set(MPI_COMM_WORLD, MPI_ERRORS_RETURN);
	int error_code;
	error_code = MPI_Comm_size(MPI_COMM_WORLD, &size);
	error_code = MPI_Comm_rank(MPI_COMM_WORLD, &rank);

	Error_Handler(error_code);
	int sendval = rank+1;
	MPI_Scan(&sendval, &fact, 1, MPI_INT, MPI_PROD, MPI_COMM_WORLD);
	fprintf(stdout, "process[%d]: fact is %d\n",rank, fact);
	fflush(stdout);
	MPI_Scan(&fact, &sum, 1, MPI_INT, MPI_SUM, MPI_COMM_WORLD);
	if(rank==size-1)
		printf("Answer is: %d\n", sum);
	return 0;
}

void Error_Handler(int error_code)
{
	if(error_code!= MPI_SUCCESS)
	{
		char error_string[BUFSIZE];
		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);
	}
```

}

Output:



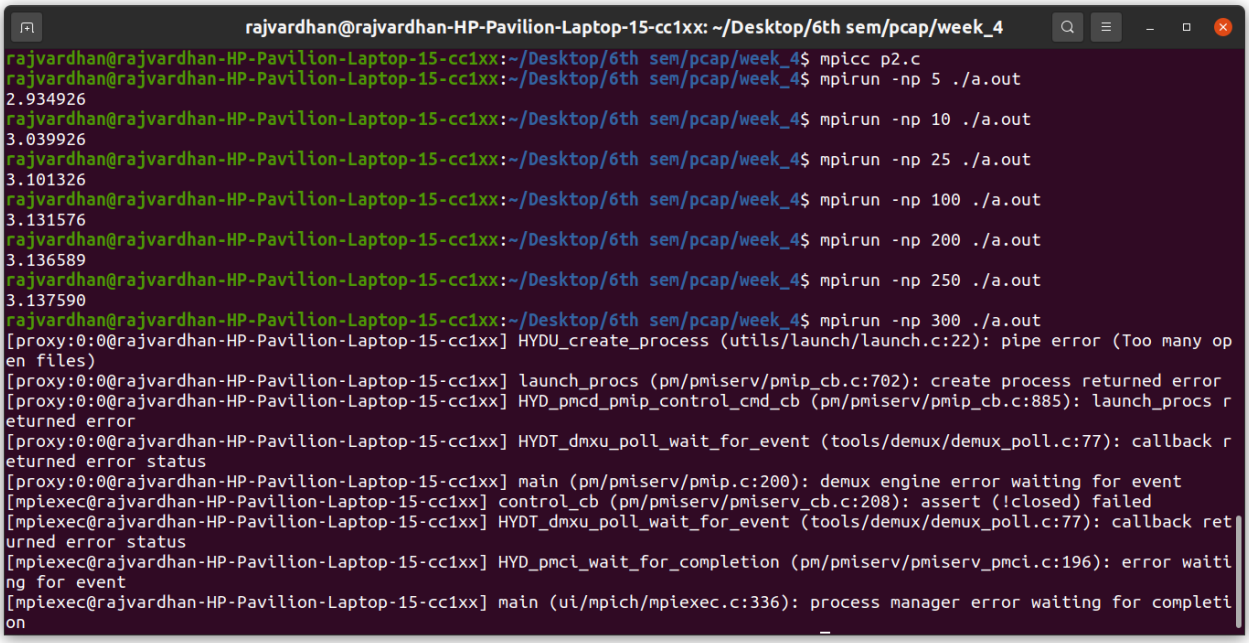
2.) Write a MPI program to calculate π-value by integrating f(x) = 4 /(1+x2). Area under the curve is divided into rectangles and the rectangles are distributed to the processors. Also handle different errors using error handling routines.

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:



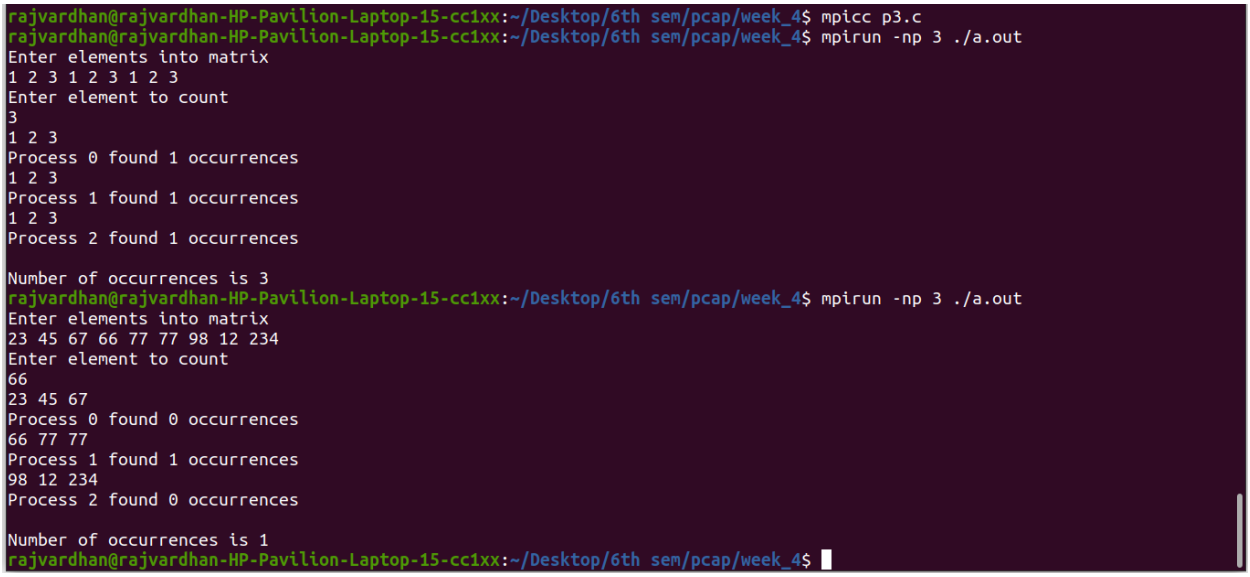
3.) Write a MPI program to read a 3 X 3 matrix. Enter an element to be searched in the root process. Find the number of occurrences of this element in the matrix using three processes.

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: