ANKITA GHOSH

180905354

Sec: A Rollno: 41

PARALLEL PROGRAMMING LAB 4

**Q1.**

Program:

#include "mpi.h"

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

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

{

int rank, size;

int i = 0;

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

int n; int error;

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);

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

{

fac = fac\*i;

}

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

if (error != MPI\_SUCCESS && rank==size-1)

{

char s[100];

int len, class1;

MPI\_Error\_string(error, s, &len);

MPI\_Error\_class(error, &class1);

printf("Error description is %s\n", s);

fflush(stderr);

printf("Error class is %d\n", class1);

fflush(stderr);

exit(0);

}

if (rank == size-1)

{

printf("Sum of 1! to %d! factorials: %d\n",size,k);

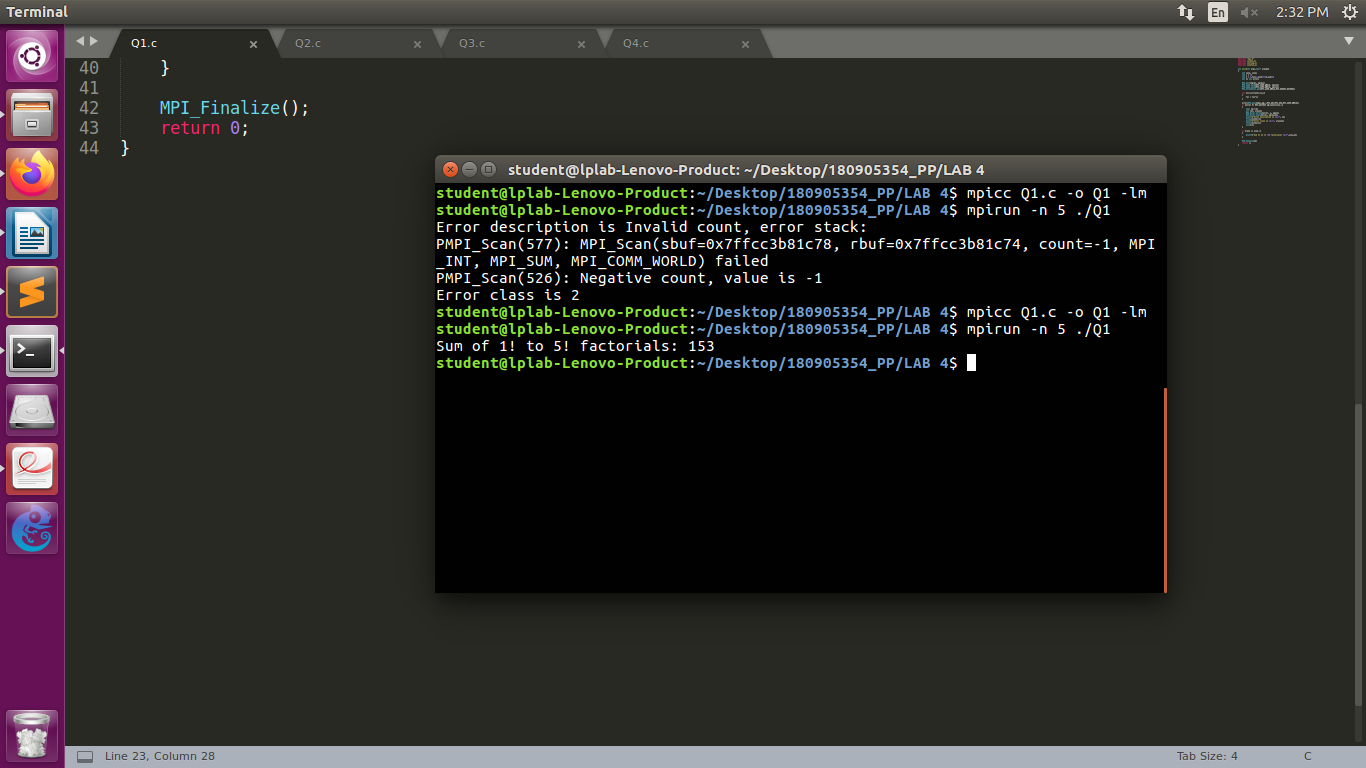
}

MPI\_Finalize();

return 0;

}

Output:



**Q2.**

Program:

#include "mpi.h"

#include <stdio.h>

#include <string.h>

#include <stdlib.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;

int error;

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);

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

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

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

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

if (error != MPI\_SUCCESS && rank==0)

{

char s[100];

int len, class1;

MPI\_Error\_string(error, s, &len);

MPI\_Error\_class(error, &class1);

printf("Error description is %s\n", s);

fflush(stderr);

printf("Error class is %d\n", class1);

fflush(stderr);

exit(0);

}

if (rank == 0)

{

printf("Area: %lf\n", pi1);

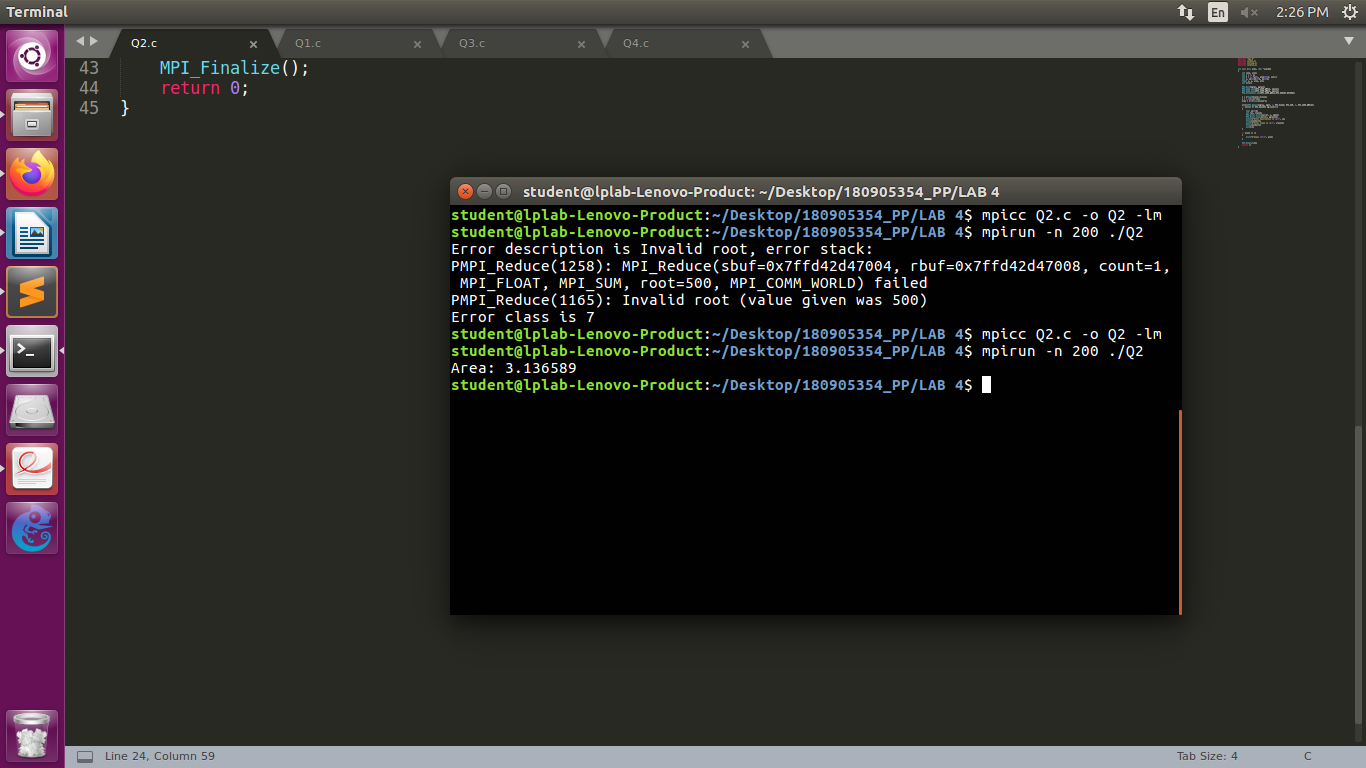
}

MPI\_Finalize();

return 0;

}

Output:



**Q3.**

Program:

#include <stdio.h>

#include "mpi.h"

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

{

int rank, ele;

int a1[3][3];

int b1[3];

int count=0;

int tc = 0;

int error;

MPI\_Init(&a, &b);

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

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

if (rank == 0)

{

printf("Enter values for 3\*3 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 be searched: \n");

scanf("%d", &ele);

}

//when root value is given as 4 it throws error

error = MPI\_Bcast(&ele, 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);

printf("Error description is %s", s);

fflush(stderr);

printf("Error class is %d", class1);

fflush(stderr);

}

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

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

{ if (b1[i] == ele)

count += 1;

}

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

if (rank == 0)

{

printf("Number of occurrences of %d= %d\n",ele,tc);

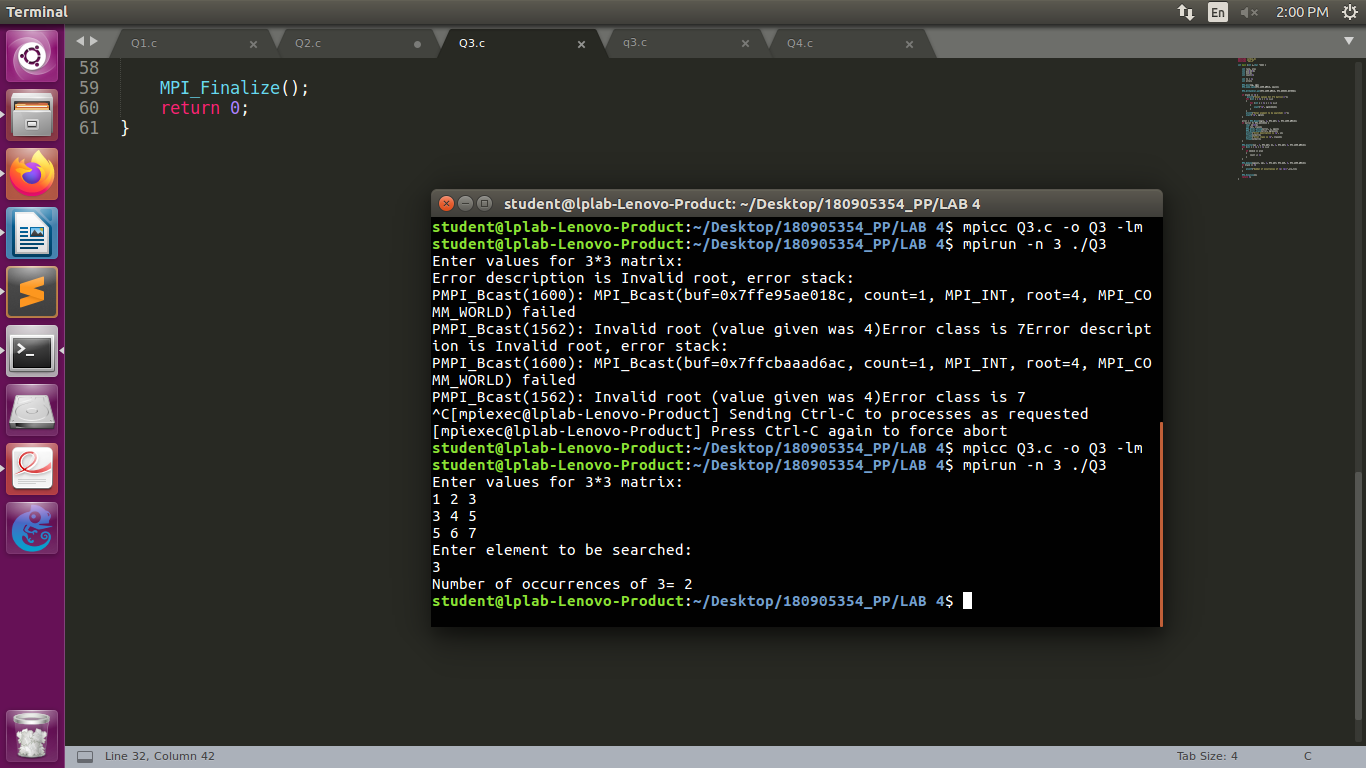
}

MPI\_Finalize();

return 0;

}

Output:



**Q4.**

Program:

#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];

MPI\_Init(&argc, &argv);

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

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("Output matrix:\n");

}

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

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

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

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

printf("\n");

MPI\_Finalize();

return 0;

}

Output:

