PPL LAB-4 EXERCISES

Date	@January 30, 2026

NAME : DEVADATHAN N R

ROLL NO : 04

SECTION : CSE A1

REG NO : 230905010

Question 1:

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

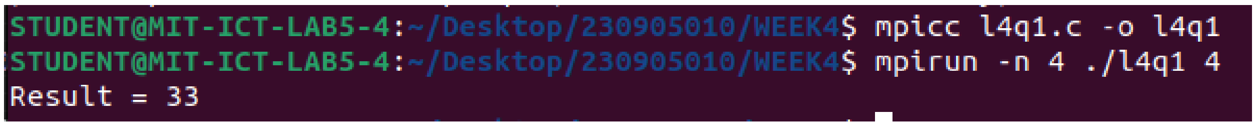
Code:

```
/*
Name : Devadathan N R
Class: CSE A1
Reg No : 230905010
Roll No : 04
*/

#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    int pid, nprocs, n;
```

```
unsigned long fact_local, fact_scan;
unsigned long sum_local, sum_global;
int status;
char err_msg[MPI_MAX_ERROR_STRING];
int msg_len;

status = MPI_Init(&argc, &argv);
if (status != MPI_SUCCESS)
{
    MPI_Error_string(status, err_msg, &msg_len);
    printf("%s\n", err_msg);
    MPI_Abort(MPI_COMM_WORLD, status);
}

MPI_Comm_rank(MPI_COMM_WORLD, &pid);
MPI_Comm_size(MPI_COMM_WORLD, &nprocs);

if (pid == 0)
{
    if (argc != 2)
    {
        printf("Usage: mpirun -n N ./a.out <N>\n");
        MPI_Abort(MPI_COMM_WORLD, 1);
    }

    n = atoi(argv[1]);

    if (n <= 0 || n != nprocs)
    {
        printf("N must be positive and equal to number of processes\n");
        MPI_Abort(MPI_COMM_WORLD, 1);
    }
}

MPI_Bcast(&n, 1, MPI_INT, 0, MPI_COMM_WORLD);
```

```
    fact_local = pid + 1;

    MPI_Scan(&fact_local, &fact_scan, 1,
             MPI_UNSIGNED_LONG, MPI_PROD, MPI_COMM_WORLD);

    sum_local = fact_scan;

    MPI_Reduce(&sum_local, &sum_global, 1,
               MPI_UNSIGNED_LONG, MPI_SUM, 0, MPI_COMM_WORLD);

    if (pid == 0)
        printf("Result = %lu\n", sum_global);

    MPI_Finalize();
    return 0;
}
```

Output:



```
STUDENT@MIT-ICT-LAB5-4:~/Desktop/230905010/WEEK4$ mpicc l4q1.c -o l4q1
STUDENT@MIT-ICT-LAB5-4:~/Desktop/230905010/WEEK4$ mpirun -n 4 ./l4q1 4
Result = 33
```

Question 2:

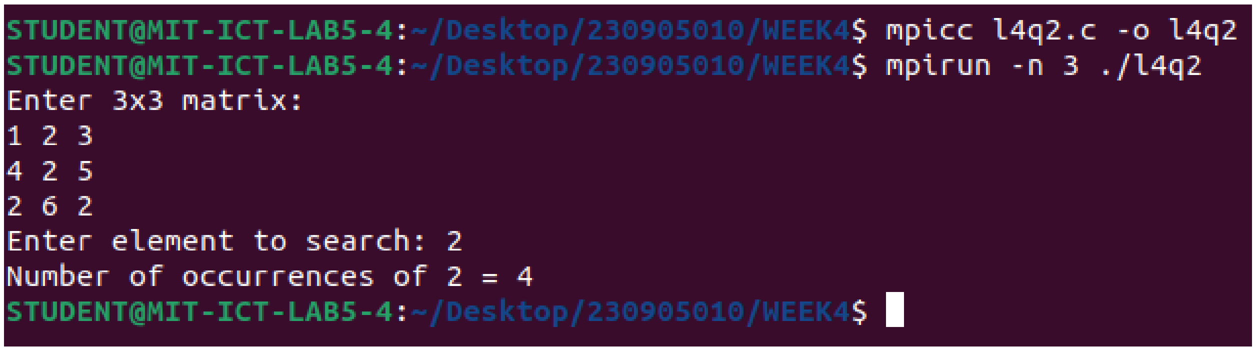
Write a MPI program to read a 3 X 3 matrix. Enter an element to be searched in the root process. Find the number of occurrences of this element in the matrix using three processes.

Code:

```
/*
Name : Devadathan N R
Class: CSE A1
```

```
Reg No : 230905010
Roll No : 04
*/

#include <stdio.h>
#include <mpi.h>

int main(int argc, char *argv[])
{
    int pid, nprocs;
    int mat[3][3];
    int key;
    int part[3];
    int count_local = 0, count_total = 0;

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &pid);
    MPI_Comm_size(MPI_COMM_WORLD, &nprocs);

    if (nprocs != 3)
    {
        if (pid == 0)
            printf("This program requires exactly 3 processes.\n");
        MPI_Finalize();
        return 0;
    }

    if (pid == 0)
    {
        printf("Enter 3×3 matrix:\n");
        for (int i = 0; i < 3; i++)
            for (int j = 0; j < 3; j++)
                scanf("%d", &mat[i][j]);

        printf("Enter element to search: ");
        scanf("%d", &key);
    }

    MPI_Bcast(&key, 1, MPI_INT, 0, MPI_COMM_WORLD);

    MPI_Scatter(mat, 3, MPI_INT,part, 3, MPI_INT,0, MPI_COMM_WORLD);

    for (int i = 0; i < 3; i++)
        if (part[i] == key)
            count_local++;

    MPI_Reduce(&count_local, &count_total, 1,MPI_INT, MPI_SUM, 0, MPI_COM
M_WORLD);

    if (pid == 0)
        printf("Number of occurrences of %d = %d\n", key, count_total);

    MPI_Finalize();
    return 0;
}
```

Output:



```
STUDENT@MIT-ICT-LAB5-4:~/Desktop/230905010/WEEK4$ mpicc l4q2.c -o l4q2
STUDENT@MIT-ICT-LAB5-4:~/Desktop/230905010/WEEK4$ mpirun -n 3 ./l4q2
Enter 3x3 matrix:
1 2 3
4 2 5
2 6 2
Enter element to search: 2
Number of occurrences of 2 = 4
STUDENT@MIT-ICT-LAB5-4:~/Desktop/230905010/WEEK4$
```

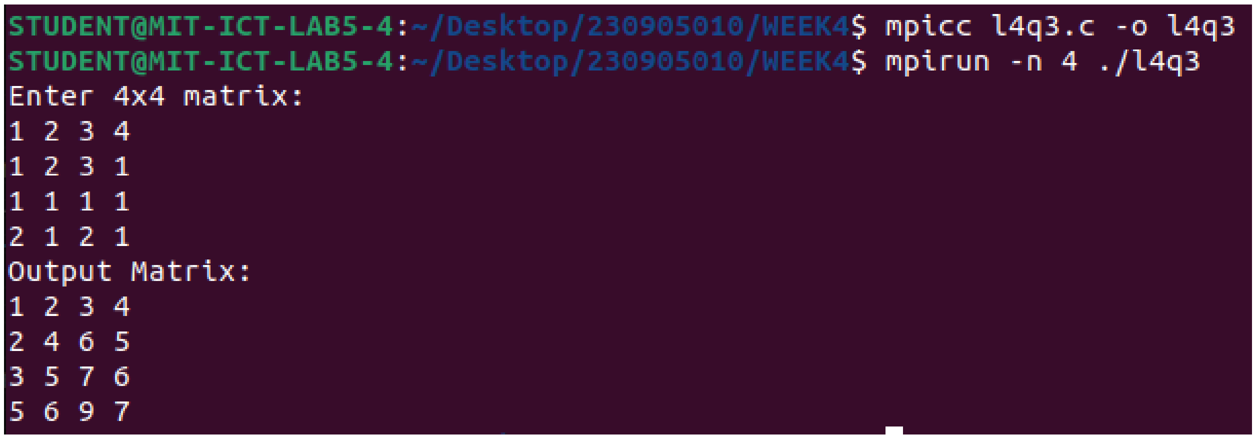
Question 3:

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

Code:

```
/*
Name : Devadathan N R
Class: CSE A1
Reg No : 230905010
Roll No : 04
*/

#include <stdio.h>
#include <mpi.h>

int main(int argc, char *argv[])
{
    int pid, nprocs;
    int mat[4][4], out[4][4];
    int block[4], prefix[4];

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &pid);
    MPI_Comm_size(MPI_COMM_WORLD, &nprocs);

    if (nprocs != 4)
    {
        if (pid == 0)
            printf("This program requires exactly 4 processes.\n");
        MPI_Finalize();
        return 0;
    }

    if (pid == 0)
    {
        printf("Enter 4×4 matrix:\n");
        for (int i = 0; i < 4; i++)
            for (int j = 0; j < 4; j++)
                scanf("%d", &mat[i][j]);
    }

    MPI_Scatter(mat, 4, MPI_INT,block, 4, MPI_INT,0, MPI_COMM_WORLD);

    MPI_Scan(block, prefix, 4, MPI_INT,MPI_SUM, MPI_COMM_WORLD);

    MPI_Gather(prefix, 4, MPI_INT,out, 4, MPI_INT,0, MPI_COMM_WORLD);

    if (pid == 0)
    {
        printf("Output Matrix:\n");
        for (int i = 0; i < 4; i++)
        {
            for (int j = 0; j < 4; j++)
                printf("%d ", out[i][j]);
            printf("\n");
        }
    }

    MPI_Finalize();
    return 0;
}
```

Output:


```
STUDENT@MIT-ICT-LAB5-4:~/Desktop/230905010/WEEK4$ mpicc l4q3.c -o l4q3
STUDENT@MIT-ICT-LAB5-4:~/Desktop/230905010/WEEK4$ mpirun -n 4 ./l4q3
Enter 4x4 matrix:
1 2 3 4
1 2 3 1
1 1 1 1
2 1 2 1
Output Matrix:
1 2 3 4
2 4 6 5
3 5 7 6
5 6 9 7
```

Question 4:

Write a MPI program to read a word of length N. Using N processes including the root get output word with the pattern as shown.

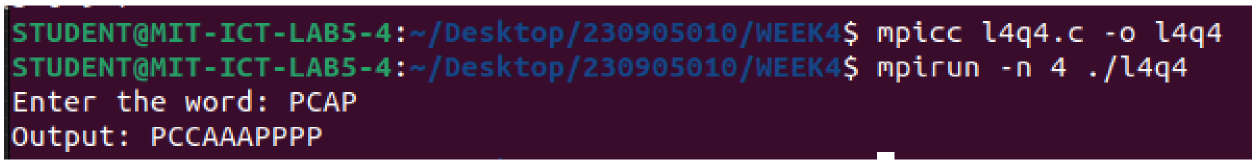
Code:

```
/*Name : Devadathan N R
Class: CSE A1
Reg No : 230905010
Roll No : 04
*/

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

int main(int argc, char *argv[])
{
   int pid, nprocs, length;
   char str[100];

   MPI_Init(&argc, &argv);
   MPI_Comm_rank(MPI_COMM_WORLD, &pid);
   MPI_Comm_size(MPI_COMM_WORLD, &nprocs);
```

```
if (pid == 0)
{
    printf("Enter the word: ");
    scanf("%s", str);
    length = strlen(str);
}

MPI_Bcast(&length, 1, MPI_INT, 0, MPI_COMM_WORLD);

if (nprocs != length)
{
    if (pid == 0)
        printf("Number of processes must be equal to word length\n");
    MPI_Finalize();
    return 0;
}

char letter;
MPI_Scatter(str, 1, MPI_CHAR,&letter, 1, MPI_CHAR,0, MPI_COMM_WORLD);

char local_buf[100];
int repeat = pid + 1;

for (int i = 0; i < repeat; i++)
    local_buf[i] = letter;
local_buf[repeat] = '\0';

int recv_cnt[100], disp[100];
int total_len = 0;

if (pid == 0)
{
    for (int i = 0; i < nprocs; i++)
    {
        recv_cnt[i] = i + 1;
```

```
            disp[i] = total_len;
            total_len += recv_cnt[i];
        }
    }

    char final_str[500];

    MPI_Gatherv(local_buf, repeat, MPI_CHAR,final_str, recv_cnt, disp,MPI_CHA
R, 0, MPI_COMM_WORLD);

    if (pid == 0)
    {
        final_str[total_len] = '\0';
        printf("Output: %s\n", final_str);
    }

    MPI_Finalize();
    return 0;
}
```

Output:



```
STUDENT@MIT-ICT-LAB5-4:~/Desktop/230905010/WEEK4$ mpicc l4q4.c -o l4q4
STUDENT@MIT-ICT-LAB5-4:~/Desktop/230905010/WEEK4$ mpirun -n 4 ./l4q4
Enter the word: PCAP
Output: PCCAAAPPPP
```