**LEBANESE AMERICAN UNIVERSITY**
**Department of Computer Science and Mathematics**
**CSC 447: Parallel Programing**
**Spring 2021**

# Lab II (MPI)

**Problem 1: Hello world**

Consider the following helloworld.c MPI program:

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

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

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);
    MPI_Get_library_version(version, &len);
    printf("Hello world, I am %d of %d, (%s, %d)\n",
           rank, size, version, len);
    MPI_Finalize();

    return 0;
}
```

This program takes an integer n as a command line argument and prints for each process the following output: *"Hello world, I am process i of n processes"*, where "i" is the rank of the process, and "n" is the size.

Compile and run this MPI program and verify its output by following these steps:

- Use an IDE of your choice to create and save your program with *YourName*_Prob1.c.
- Compile your program within the IDE or from terminal window by using the command: mpicc –o Problem1 YourName_Prob1.c
- Then, run your MPI program by typing the command in a terminal widow: mpirun -n <NumCores> <compiledProgram>
- Verify that your program gives the correct output for 1, 2, and 4 cores

**Problem 2: Average of an array of numbers**

Write a parallel program that takes, as a command line argument, the name of a file and reads from it an integer n followed by n float integers. Your program must compute the average of the array and print a report to an output file.

**Sample output report (using 4 processes):**
Process 0: local average = ….
Process 1: local average = ….
Process 2: local average = ….
Process 3: local average = ….
Global average = ….

- Test your program on the provided text file *Input1.txt*
- Measure the execution time in your program using the function $MPI\_Wtime()$. This can be done as follows:

```
Double start_time = MPI_Wtime();

//Insert your parallel code here

if(rank==0)
    {
       double end_time=MPI_Wtime();
       printf("Wallclock time elapsed: %.8lf seconds\n",end_time-
start_time);
    }
```

- Report in a tabular form the execution time of your program when using 1, 2, and 4 processors.
- Draw your conclusions on the speedups obtained.

Save your program with the name *YourName*_Prob2.c.

*Note: A report should be submitted where you describe your parallel implementation and report in, tabular form, the execution times and speedups obtained.*

***Problem 3: Matrix Multiplication***

Write a parallel program that takes, as a command line argument, the name of a file and reads from it first two integers that correspond to the dimensions (number of columns and number of rows) of the first matrix followed by the elements of the matrix, then two integers that correspond to the dimensions of the second matrix followed by the elements of the second matrix. The two matrices are read into two 2-dimensional arrays. Your program should compute their multiplication and write the result to an output file. **Note**: you must handle the case when the matrices cannot be multiplied by printing "No answer" in this case.

| **Sample input:** | **Sample output:** |
|---|---|
| 3 | 17 17 |

```
4
1 2 5 0
3 0 1 4
5 2 2 3
4
2
0 2
1 5
3 1
1 6
```

```
7   31
11  40
```

- Test your program on the provided text file *Input2.txt*
- Measure the execution time in your program using the function $MPI\_Wtime$ as was explained in Problem 1
- Report in a tabular form the execution time of your program when using 1, 2, and 4 processors.
- Draw your conclusions on the speedups obtained.

*Note: A report should be submitted where you describe your parallel implementation for every problem and report in, tabular form, the execution*

**Submission Instructions:**

You are required to submit your solution to Problem 2 today by the end of this session. Place your code and report in one folder and submit on the provided submission link on Blackboard.

You are required to submit your solution to **Problem 3 by Tuesday March 2 8:00 am**.. Place your code and report in one folder and submit on the provided submission link on Blackboard.