**Name: Varun Menon**

**Reg No: 19BCE1438**

**Course Code: CSE4001**

**Faculty: Dr. Harini S**

**Lab Experiment 8**

1. Sample Hello World

Code:

#include<stdio.h>

#include <mpi.h>

int main()

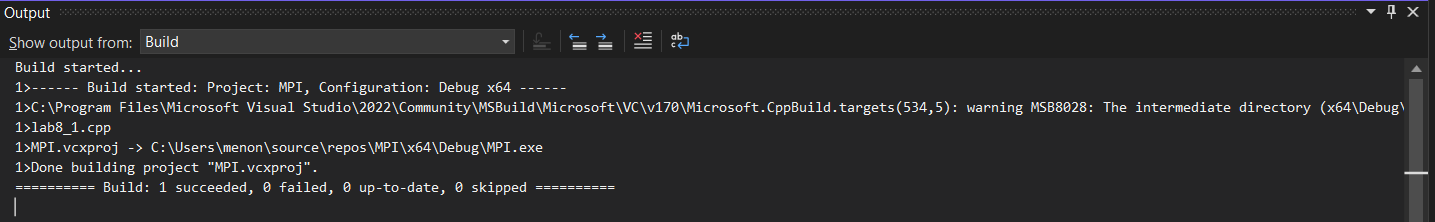
{

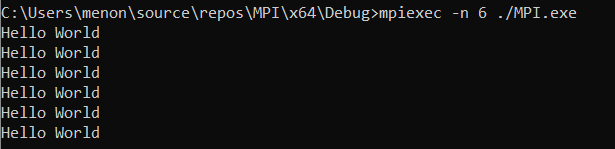
    printf("Hello World\n");

    return 0;

}

Output:





1. Prink rank, world size and processor name

Code:

#include <mpi.h>

#include <stdio.h>

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

{

    MPI\_Init(NULL, NULL);

    int world\_size;

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

    printf("World Size: %d\n", world\_size);

    int world\_rank;

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

    char processor\_name[MPI\_MAX\_PROCESSOR\_NAME];

    int name\_len;

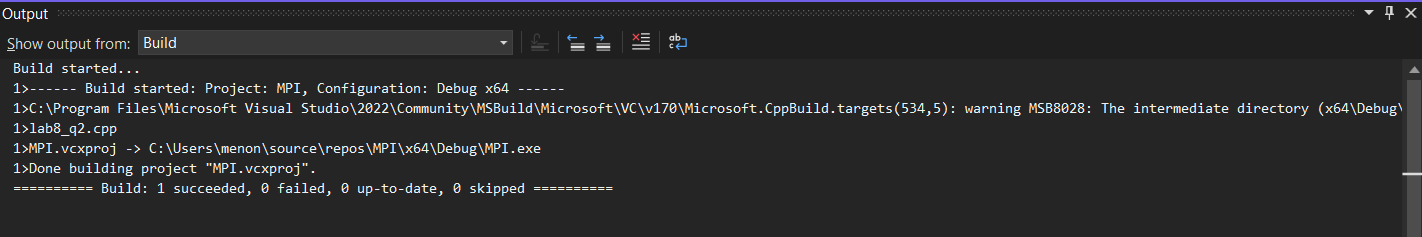
    MPI\_Get\_processor\_name(processor\_name, &name\_len);

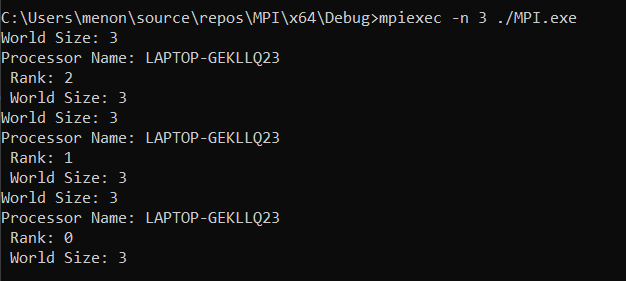
    printf("Processor Name: %s\n Rank: %d\n World Size: %d \n", processor\_name, world\_rank, world\_size);

    MPI\_Finalize();

}

Output:





1. Master prints "I am Master", Worker prints "I am worker"

Code:

#include <stdio.h>

#include <omp.h>

int main()

{

    int n = 3;

    int a[3][3] = {

        {1,2,3},

        {4,5,6},

        {7,8,9}

    };

    for(int i=1;i<n;i++)

    {

        #pragma omp parallel for

        for(int j=0;j<n;j++)

        {

            a[i][j] = a[i-1][j] + 2;

        }

    }

    for(int i=0;i<n;i++)

    {

        for(int j=0;j<n;j++)

        {

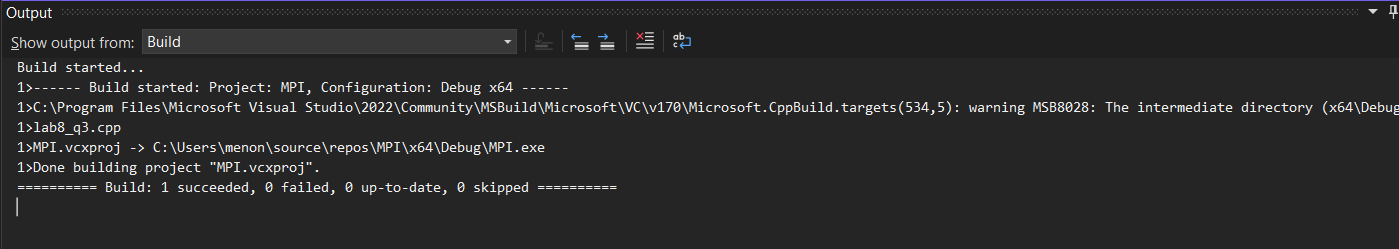
            printf("a[%d][%d] = %d\n", i, j, a[i][j]);

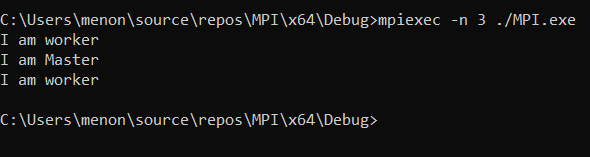
        }

    }

}

Output:





1. Master generates 1/2,1/4,1/8,1/16...1/n; Worker generates 2,4,8,16...n

Code:

#include <mpi.h>

#include <stdio.h>

int rank, numprocs, left, right, n;

MPI\_Request request, request2;

MPI\_Status status;

void slave\_method()

{

    double buffer2[10];

    buffer2[0] = 2;

    for (int i = 1; i < n; i++)

    {

        buffer2[i] = buffer2[i - 1] \* 2;

    }

    MPI\_Isend(buffer2, n, MPI\_DOUBLE, right, 123, MPI\_COMM\_WORLD, &request2);

    MPI\_Wait(&request2, &status);

    printf("By slave: ");

    for (int i = 0; i < n; i++)

    {

        printf("%f ", buffer2[i]);

    }

    printf("\n");

}

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

{

    MPI\_Init(&argc, &argv);

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

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

    right = (rank + 1) % numprocs;

    left = rank - 1;

    if (left < 0)

        left = numprocs - 1;

    n = 10;

    double buffer[10];

    slave\_method();

    MPI\_Irecv(buffer, n, MPI\_DOUBLE, left, 123, MPI\_COMM\_WORLD, &request);

    MPI\_Wait(&request, &status);

    MPI\_Wait(&request2, &status);

    MPI\_Finalize();

    if (rank == 0)

    {

        printf("From slave: ");

        for (int i = 0; i < n; i++)

        {

            printf("%f ", buffer[i]);

        }

        printf("\n");

        printf("By master: ");

        for (int i = 0; i < n; i++)

        {

            double value = 1.0 / buffer[i];

            printf("%f ", value);

        }

        printf("\n");

    }

    return 0;

}

Output:

