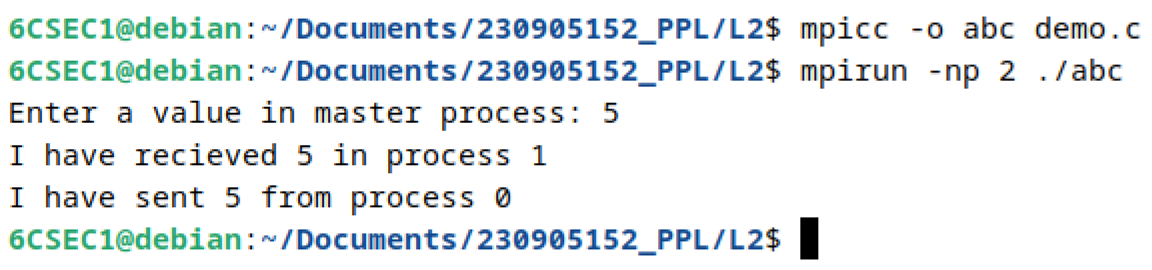
Lab2- Point to Point Communications in MPI

SolvedQ.

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

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

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    if (rank == 0) {
        printf("Enter a value in master process: ");
        fflush(stdout);
        scanf("%d", &x);

        MPI_Send(&x, 1, MPI_INT, 1, 1, MPI_COMM_WORLD);

        fprintf(stdout, "I have sent %d from process 0\n", x);
        fflush(stdout);
    }
    else if (rank == 1) {
        MPI_Recv(&x, 1, MPI_INT, 0, 1, MPI_COMM_WORLD, &status);
        fprintf(stdout, "I have received %d in process 1\n", x);
        fflush(stdout);
    }

    MPI_Finalize();
    return 0;
}
```

Screenshot:

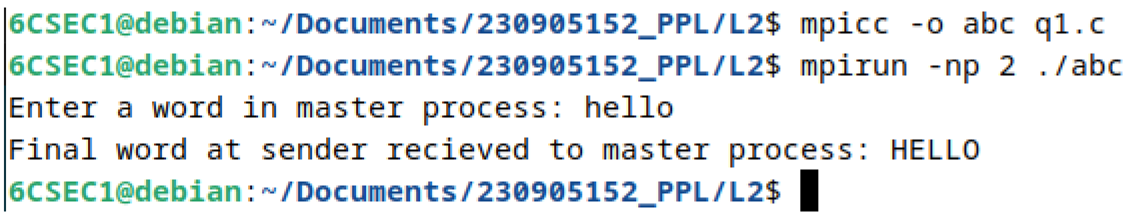


```
6CSEC1@debian:~/Documents/230905152_PPL/L2$ mpicc -o abc demo.c
6CSEC1@debian:~/Documents/230905152_PPL/L2$ mpirun -np 2 ./abc
Enter a value in master process: 5
I have recieved 5 in process 1
I have sent 5 from process 0
6CSEC1@debian:~/Documents/230905152_PPL/L2$ 
```

Q1.

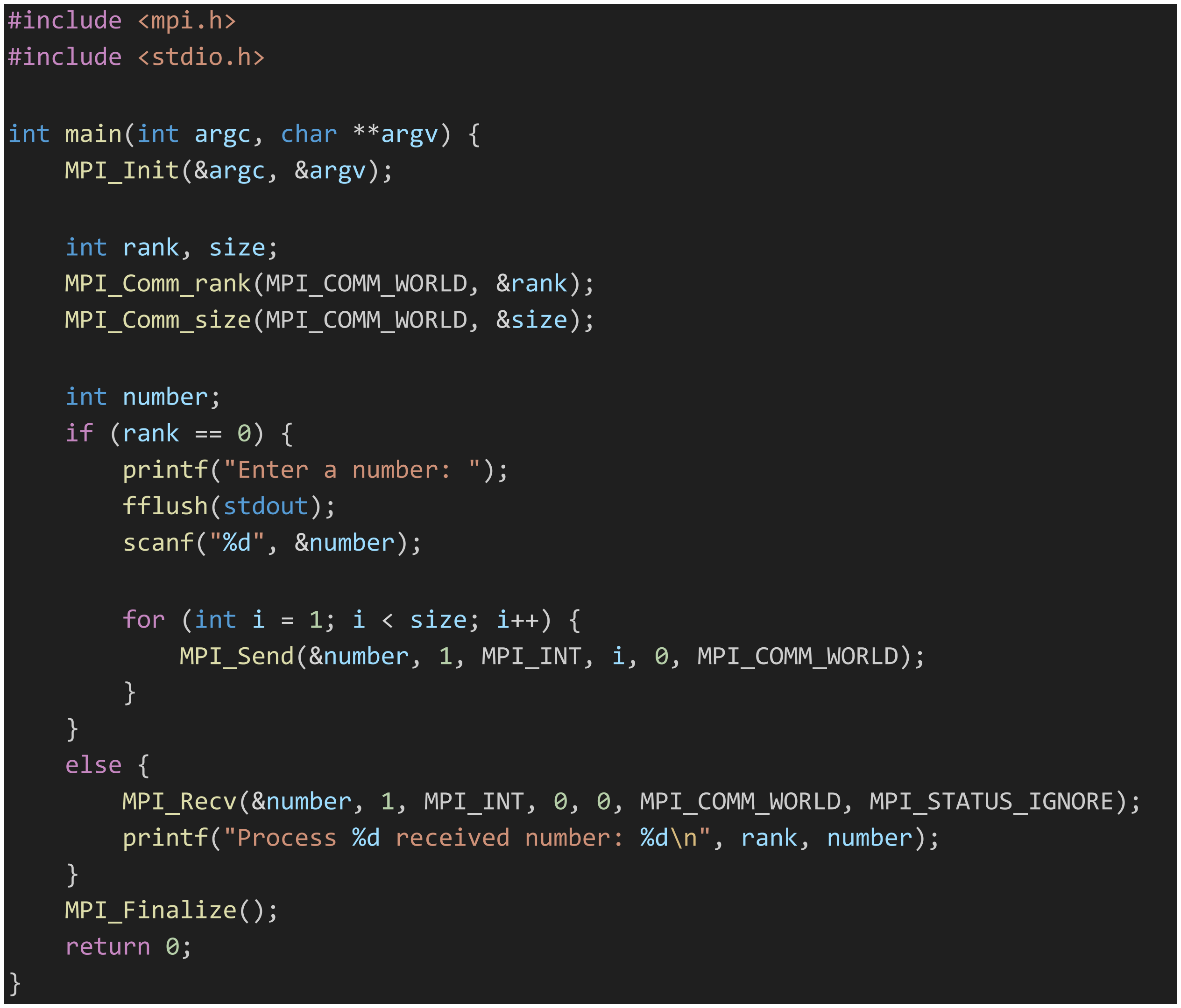
```c
#include <mpi.h>
#include <stdio.h>
#include <ctype.h>
#include <string.h>

int main(int argc, char **argv) {
    MPI_Init(&argc, &argv);

    int rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    char word[50];

    if (rank == 0) {
        printf("Enter a word in master process: ");
        fflush(stdout);
        scanf("%s", word);

        MPI_Ssend(word, strlen(word) + 1, MPI_CHAR, 1, 10, MPI_COMM_WORLD);
        MPI_Recv(word, 50, MPI_CHAR, 1, 20, MPI_COMM_WORLD, MPI_STATUS_IGNORE);

        printf("Final word at sender: %s\n", word);
    }
    else if (rank == 1) {
        MPI_Recv(word, 50, MPI_CHAR, 0, 10, MPI_COMM_WORLD, MPI_STATUS_IGNORE);

        for (int i = 0; word[i]; i++) {
            if (islower(word[i])) {
                word[i] = word[i] - 32;
            } else if (isupper(word[i])) {
                word[i] = word[i] + 32;
            }
        }

        MPI_Ssend(word, strlen(word) + 1, MPI_CHAR, 0, 20, MPI_COMM_WORLD);
    }

    MPI_Finalize();
    return 0;
}
```

Screenshot:



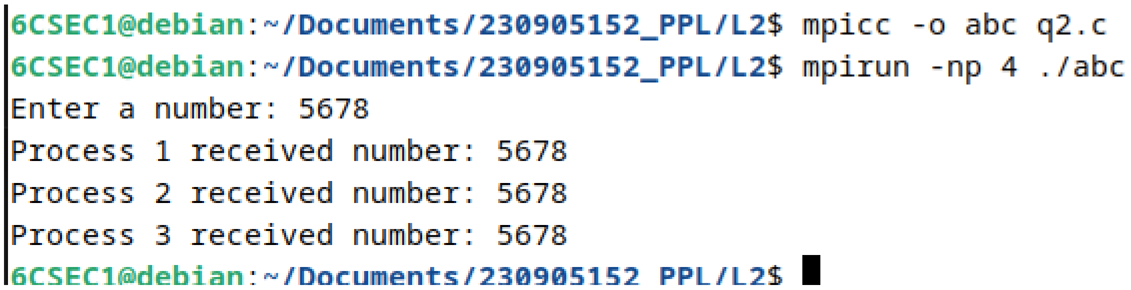
```
6CSEC1@debian:~/Documents/230905152_PPL/L2$ mpicc -o abc q1.c
6CSEC1@debian:~/Documents/230905152_PPL/L2$ mpirun -np 2 ./abc
Enter a word in master process: hello
Final word at sender recieved to master process: HELLO
6CSEC1@debian:~/Documents/230905152_PPL/L2$ 
```

Q2.



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

int main(int argc, char **argv) {
    MPI_Init(&argc, &argv);

    int rank, size;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    int number;
    if (rank == 0) {
        printf("Enter a number: ");
        fflush(stdout);
        scanf("%d", &number);

        for (int i = 1; i < size; i++) {
            MPI_Send(&number, 1, MPI_INT, i, 0, MPI_COMM_WORLD);
        }
    }
    else {
        MPI_Recv(&number, 1, MPI_INT, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
        printf("Process %d received number: %d\n", rank, number);
    }
    MPI_Finalize();
    return 0;
}
```

Screenshot:

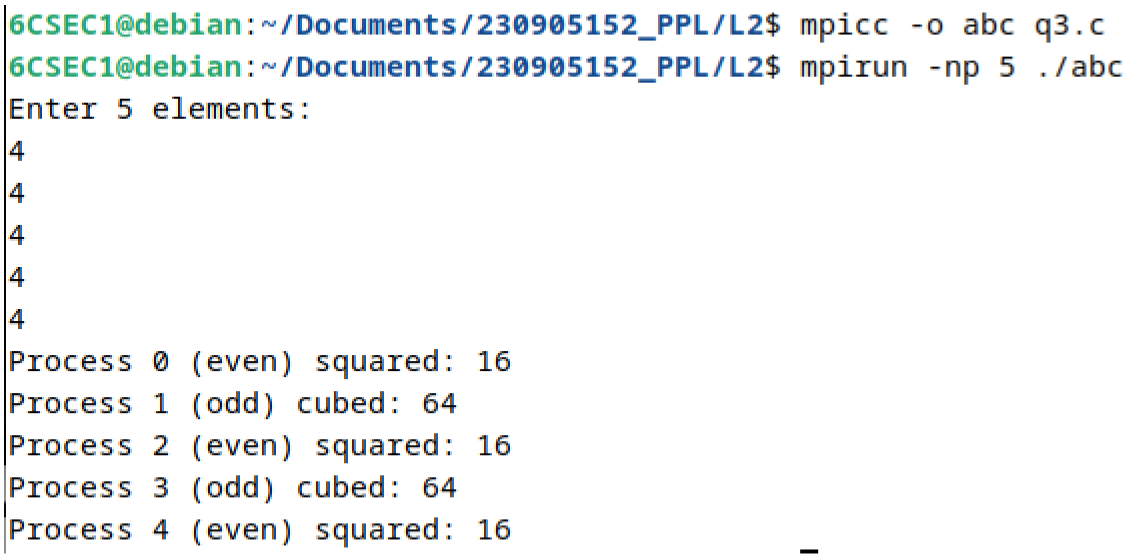


```
6CSEC1@debian:~/Documents/230905152_PPL/L2$ mpicc -o abc q2.c
6CSEC1@debian:~/Documents/230905152_PPL/L2$ mpirun -np 4 ./abc
Enter a number: 5678
Process 1 received number: 5678
Process 2 received number: 5678
Process 3 received number: 5678
6CSEC1@debian:~/Documents/230905152_PPL/L2$ 
```

Q3.

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

int main(int argc, char **argv) {
    MPI_Init(&argc, &argv);

    int rank, size;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    int *arr = NULL;
    int value;

    int bufsize = size * (sizeof(int) + MPI_BSEND_OVERHEAD);
    void *buffer = malloc(bufsize);
    MPI_Buffer_attach(buffer, bufsize);

    if (rank == 0) {
        arr = (int *)malloc(size * sizeof(int));
        printf("Enter %d elements:\n", size);
        fflush(stdout);
        for (int i = 0; i < size; i++) {
            scanf("%d", &arr[i]);
        }
        for (int i = 1; i < size; i++) {
            MPI_Bsend(&arr[i], 1, MPI_INT, i, 0, MPI_COMM_WORLD);
        }
        value = arr[0];
        printf("Process %d (even) squared: %d\n", rank, value * value);
        free(arr);
    }
    else {
        MPI_Recv(&value, 1, MPI_INT, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
        if (rank % 2 == 0) {
            printf("Process %d (even) squared: %d\n", rank, value * value);
        }
        else {
            printf("Process %d (odd) cubed: %d\n", rank, value * value * value);
        }
    }
    MPI_Buffer_detach(&buffer, &bufsize);
    free(buffer);
    MPI_Finalize();
    return 0;
}
```

Screenshot:

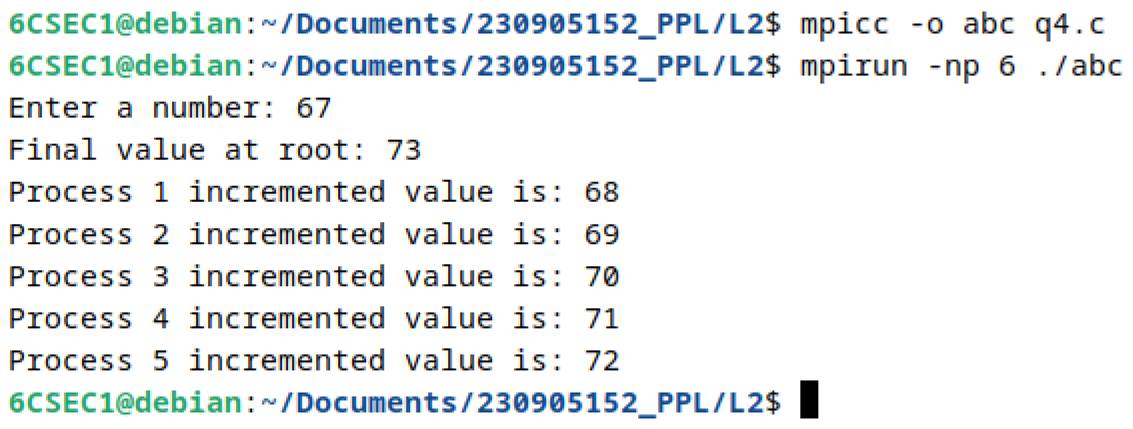


```
6CSEC1@debian:~/Documents/230905152_PPL/L2$ mpicc -o abc q3.c
6CSEC1@debian:~/Documents/230905152_PPL/L2$ mpirun -np 5 ./abc
Enter 5 elements:
4
4
4
4
4
Process 0 (even) squared: 16
Process 1 (odd) cubed: 64
Process 2 (even) squared: 16
Process 3 (odd) cubed: 64
Process 4 (even) squared: 16
```

Q4.

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

int main(int argc, char **argv) {
    MPI_Init(&argc, &argv);

    int rank, size;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    int value;
    if (rank == 0) {
        printf("Enter a number: ");
        fflush(stdout);
        scanf("%d", &value);

        MPI_Send(&value, 1, MPI_INT, 1, 0, MPI_COMM_WORLD);
        MPI_Recv(&value, 1, MPI_INT, size - 1, 0, MPI_COMM_WORLD,
MPI_STATUS_IGNORE);

        printf("Final value at root: %d\n", value);
    }
    else {
        MPI_Recv(&value, 1, MPI_INT, rank - 1, 0, MPI_COMM_WORLD,
MPI_STATUS_IGNORE);
        value++;

        if (rank == size - 1) {
            MPI_Send(&value, 1, MPI_INT, 0, 0, MPI_COMM_WORLD);
        }
        else {
            MPI_Send(&value, 1, MPI_INT, rank + 1, 0, MPI_COMM_WORLD);
        }
    }
    for (int i = 1; i < size; i++) {
        MPI_Barrier(MPI_COMM_WORLD);
        if (rank == i) {
            printf("Process %d incremented value is: %d\n", rank, value);
            fflush(stdout);
        }
    }
    MPI_Finalize();
    return 0;
}
```

Screenshot:

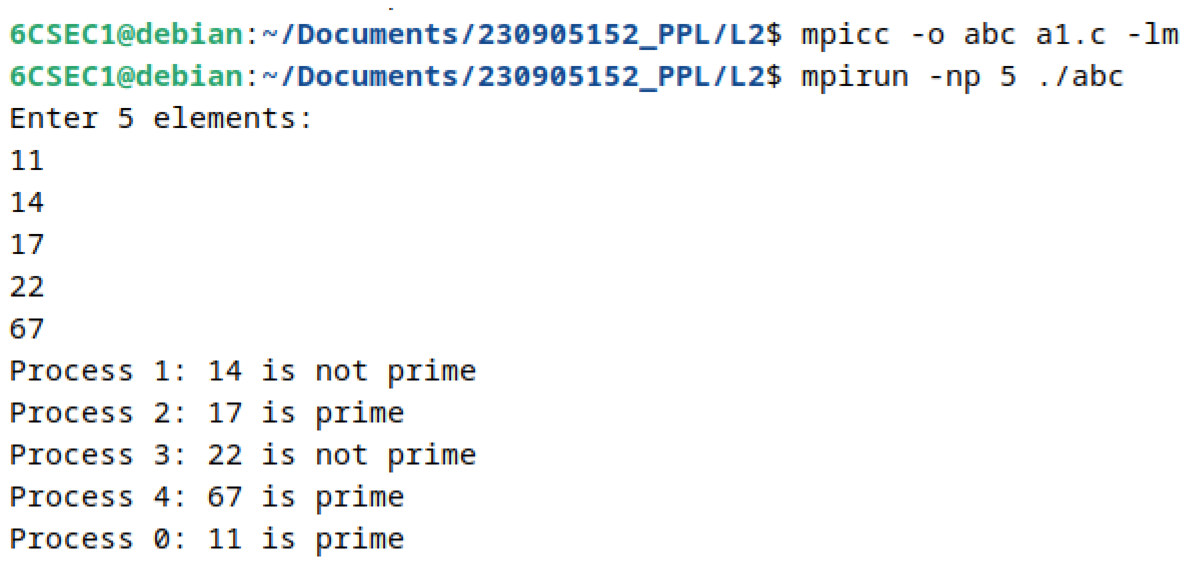


```
6CSEC1@debian:~/Documents/230905152_PPL/L2$ mpicc -o abc q4.c
6CSEC1@debian:~/Documents/230905152_PPL/L2$ mpirun -np 6 ./abc
Enter a number: 67
Final value at root: 73
Process 1 incremented value is: 68
Process 2 incremented value is: 69
Process 3 incremented value is: 70
Process 4 incremented value is: 71
Process 5 incremented value is: 72
6CSEC1@debian:~/Documents/230905152_PPL/L2$
```

A1.

```
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <stdbool.h>
bool isPrime(int num) {
    if (num <= 1) return 0;
    if (num == 2) return 1;
    if (num % 2 == 0) return 0;
    for (int i = 3; i <= sqrt(num); i += 2) {
        if (num % i == 0) return 0;
    }
    return 1;
}
int main(int argc, char **argv) {
    MPI_Init(&argc, &argv);

    int rank, size;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    int *arr = NULL;
    int value;
    if (rank == 0) {
        arr = (int *)malloc(size * sizeof(int));
        printf("Enter %d elements:\n", size);
        for (int i = 0; i < size; i++) {
            scanf("%d", &arr[i]);
        }
        for (int i = 1; i < size; i++) {
            MPI_Send(&arr[i], 1, MPI_INT, i, 0, MPI_COMM_WORLD);
        }
        value = arr[0];
        if (isPrime(value))
            printf("Process %d: %d is prime\n", rank, value);
        else
            printf("Process %d: %d is not prime\n", rank, value);

        free(arr);
    }
    else {
        MPI_Recv(&value, 1, MPI_INT, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
        if (isPrime(value))
            printf("Process %d: %d is prime\n", rank, value);
        else
            printf("Process %d: %d is not prime\n", rank, value);
    }
    MPI_Finalize();
    return 0;
}
```

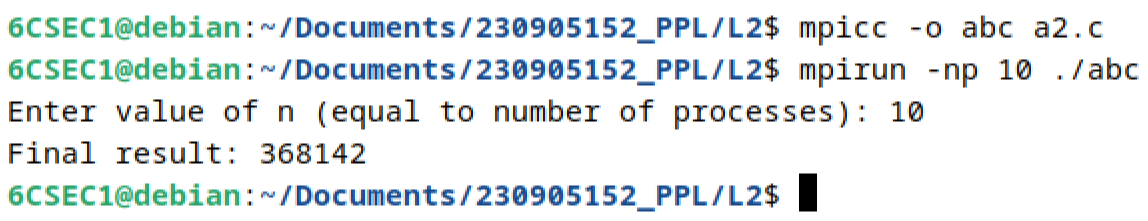
Screenshot:



```
6CSEC1@debian:~/Documents/230905152_PPL/L2$ mpicc -o abc a1.c -lm
6CSEC1@debian:~/Documents/230905152_PPL/L2$ mpirun -np 5 ./abc
Enter 5 elements:
11
14
17
22
67
Process 1: 14 is not prime
Process 2: 17 is prime
Process 3: 22 is not prime
Process 4: 67 is prime
Process 0: 11 is prime
```

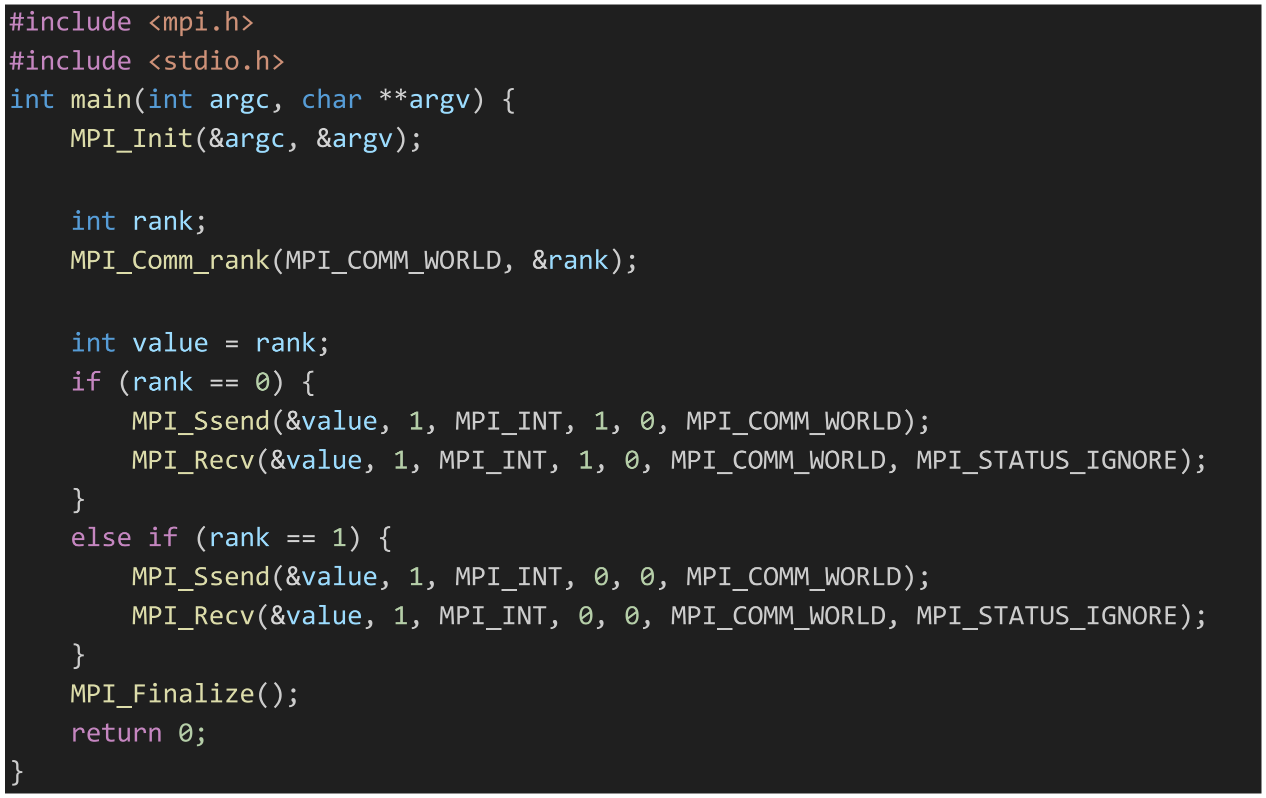
A2.

```c
#include <mpi.h>
#include <stdio.h>
int factorial(int num) {
    int fact = 1;
    for (int i = 1; i <= num; i++) {
        fact *= i;
    }
    return fact;
}
int sumSeries(int k) {
    int sum = 0;
    for (int i = 1; i <= k; i++) {
        sum += i;
    }
    return sum;
}
int main(int argc, char **argv) {
    MPI_Init(&argc, &argv);

    int rank, size;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    int n;
    int local_result = 0, final_result = 0;
    if (rank == 0) {
        printf("Enter value of n (equal to number of processes): ");
        fflush(stdout);
        scanf("%d", &n);

        if (n != size) {
            printf("Error: run with -np %d processes\n", n);
            MPI_Abort(MPI_COMM_WORLD, 1);
        }
        for (int i = 1; i < size; i++) {
            MPI_Send(&n, 1, MPI_INT, i, 0, MPI_COMM_WORLD);
        }
    }
    else {
        MPI_Recv(&n, 1, MPI_INT, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
    }

    if (rank % 2 == 0) {
        local_result = factorial(rank + 1);
    } else {
        local_result = sumSeries(rank + 1);
    }

    if (rank == 0) {
        final_result = local_result;
        int temp_res;
        for (int i = 1; i < size; i++) {
            MPI_Recv(&temp_res, 1, MPI_INT, i, 1, MPI_COMM_WORLD,
MPI_STATUS_IGNORE);
            final_result += temp_res;
        }
        printf("Final result: %d\n", final_result);
    }
    else {
        MPI_Send(&local_result, 1, MPI_INT, 0, 1, MPI_COMM_WORLD);
    }

    MPI_Finalize();
    return 0;
}
```
Screenshot:



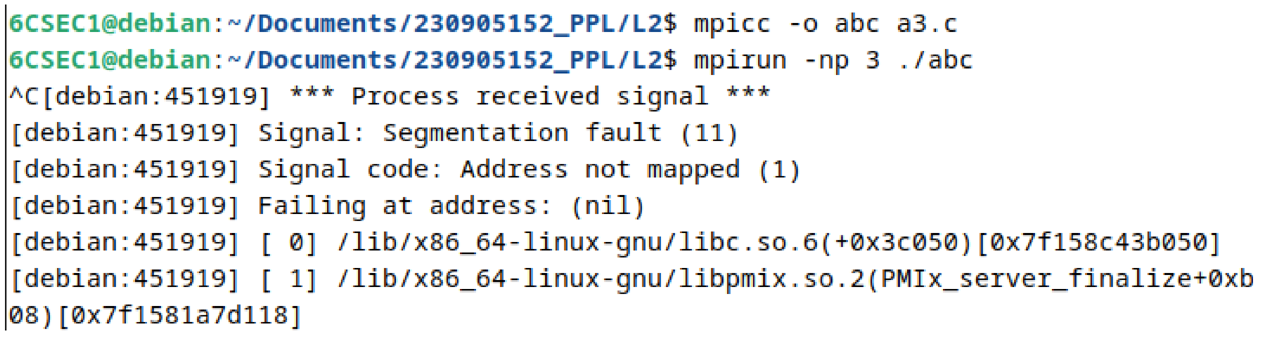
```
6CSEC1@debian:~/Documents/230905152_PPL/L2$ mpicc -o abc a2.c
6CSEC1@debian:~/Documents/230905152_PPL/L2$ mpirun -np 10 ./abc
Enter value of n (equal to number of processes): 10
Final result: 368142
6CSEC1@debian:~/Documents/230905152_PPL/L2$ █
```

A3.



```c
#include <mpi.h>
#include <stdio.h>
int main(int argc, char **argv) {
    MPI_Init(&argc, &argv);

    int rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);

    int value = rank;
    if (rank == 0) {
        MPI_Ssend(&value, 1, MPI_INT, 1, 0, MPI_COMM_WORLD);
        MPI_Recv(&value, 1, MPI_INT, 1, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
    }
    else if (rank == 1) {
        MPI_Ssend(&value, 1, MPI_INT, 0, 0, MPI_COMM_WORLD);
        MPI_Recv(&value, 1, MPI_INT, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
    }
    MPI_Finalize();
    return 0;
}
```

Screenshot:



```
6CSEC1@debian:~/Documents/230905152_PPL/L2$ mpicc -o abc a3.c
6CSEC1@debian:~/Documents/230905152_PPL/L2$ mpirun -np 3 ./abc
^C[debian:451919] *** Process received signal ***
[debian:451919] Signal: Segmentation fault (11)
[debian:451919] Signal code: Address not mapped (1)
[debian:451919] Failing at address: (nil)
[debian:451919] [ 0] /lib/x86_64-linux-gnu/libc.so.6(+0x3c050)[0x7f158c43b050]
[debian:451919] [ 1] /lib/x86_64-linux-gnu/libpmix.so.2(PMIx_server_finalize+0xb
08)[0x7f1581a7d118]
```