CSE4001 - Parallel and Distributed Computing

Lab 21+22

Digital Assignment- 5

Submitted by: Alokam Nikhitha

Reg No:19BCE2555

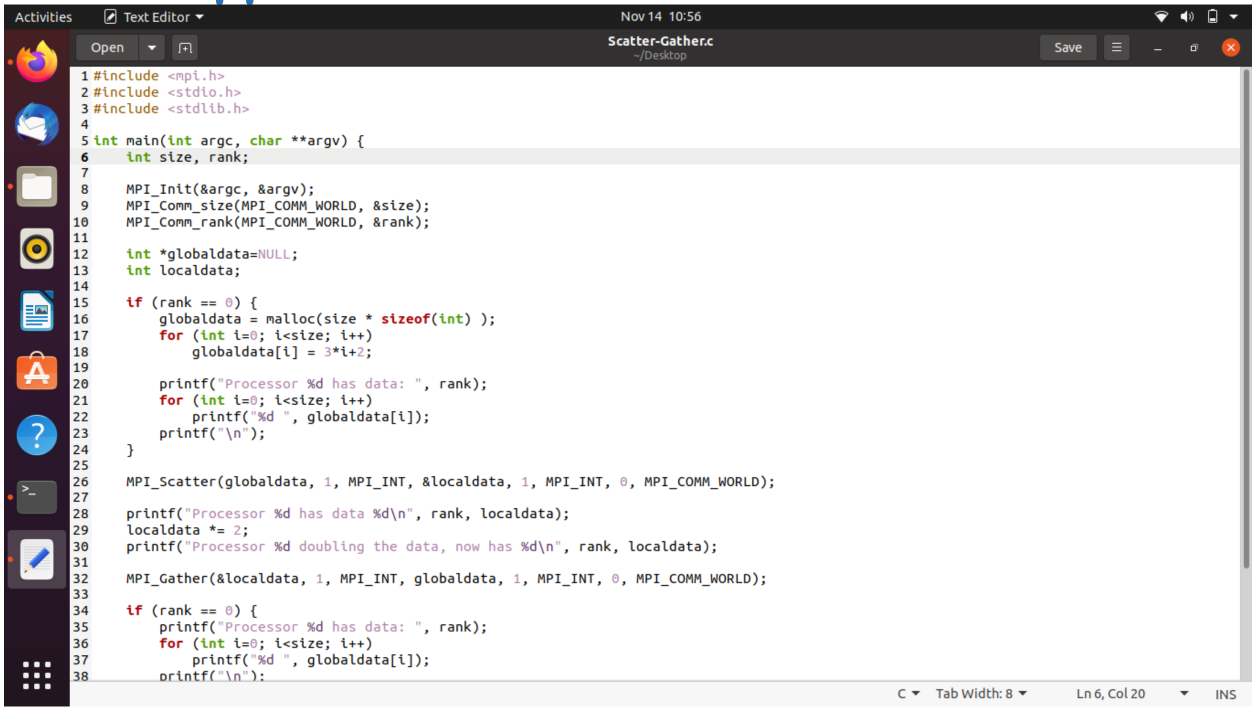
QUESTION:

Write a C program to handle message passing in the MPI application interface using Group Operators: Scatter and Gather.

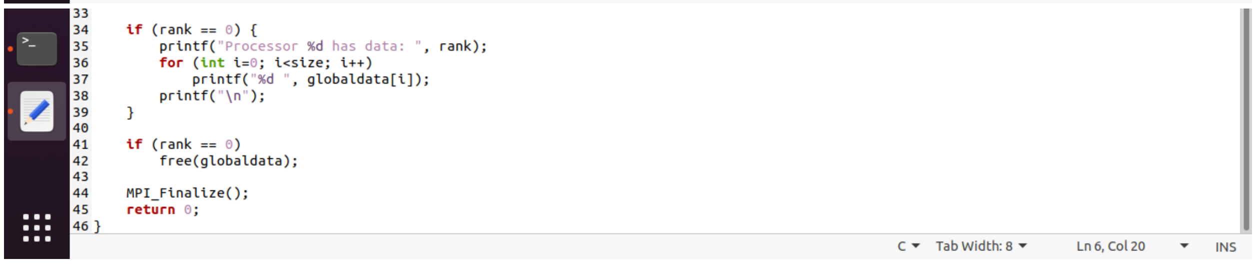
CODE:

```
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char **argv) {
   int size, rank;
   MPI_Init(&argc, &argv);
   MPI_Comm_size(MPI_COMM_WORLD, &size);
   MPI_Comm_rank(MPI_COMM_WORLD, &rank);
   int *globaldata=NULL;
   int localdata;
   if (rank == 0) {
      globaldata = malloc(size * sizeof(int) );
      for (int i=0; i<size; i++)
         globaldata[i] = 3*i+2;
      printf("Processor %d has data: ", rank);
      for (int i=0; i<size; i++)
         printf("%d ", globaldata[i]);
      printf("\n");
   }
   MPI_Scatter(globaldata, 1, MPI_INT, &localdata, 1, MPI_INT, 0,
MPI_COMM_WORLD);
   printf("Processor %d has data %d\n", rank, localdata);
   localdata *= 2;
   printf("Processor %d doubling the data, now has %d\n", rank, localdata);

   MPI_Gather(&localdata, 1, MPI_INT, globaldata, 1, MPI_INT, 0,
MPI_COMM_WORLD);
   if (rank == 0) {
      printf("Processor %d has data: ", rank);
      for (int i=0; i<size; i++)
         printf("%d ", globaldata[i]);
      printf("\n");
   }
   if (rank == 0)
      free(globaldata);
   MPI_Finalize();
   return 0;
}
```

Code Snippets:

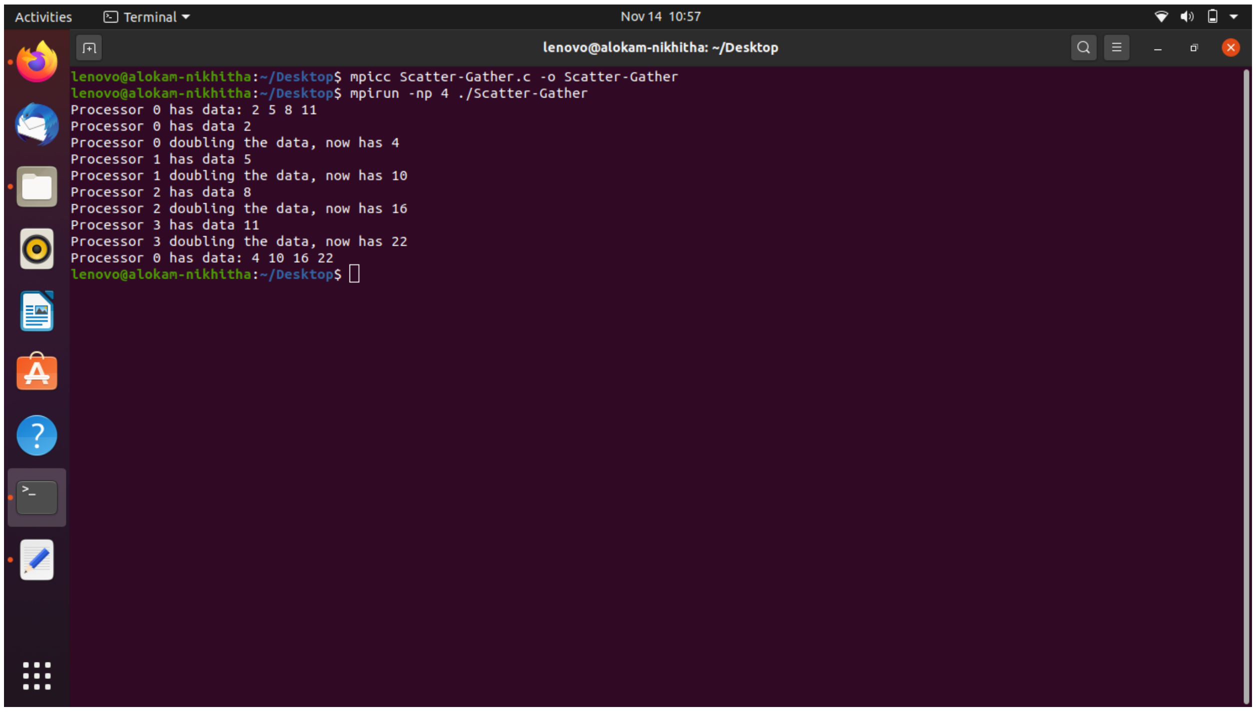


```c
#include <mpi.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char **argv) {
    int size, rank;

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

    int *globaldata=NULL;
    int localdata;

    if (rank == 0) {
        globaldata = malloc(size * sizeof(int) );
        for (int i=0; i<size; i++)
            globaldata[i] = 3*i+2;

        printf("Processor %d has data: ", rank);
        for (int i=0; i<size; i++)
            printf("%d ", globaldata[i]);
        printf("\n");
    }

    MPI_Scatter(globaldata, 1, MPI_INT, &localdata, 1, MPI_INT, 0, MPI_COMM_WORLD);

    printf("Processor %d has data %d\n", rank, localdata);
    localdata *= 2;
    printf("Processor %d doubling the data, now has %d\n", rank, localdata);

    MPI_Gather(&localdata, 1, MPI_INT, globaldata, 1, MPI_INT, 0, MPI_COMM_WORLD);

    if (rank == 0) {
        printf("Processor %d has data: ", rank);
        for (int i=0; i<size; i++)
            printf("%d ", globaldata[i]);
        printf("\n");
```



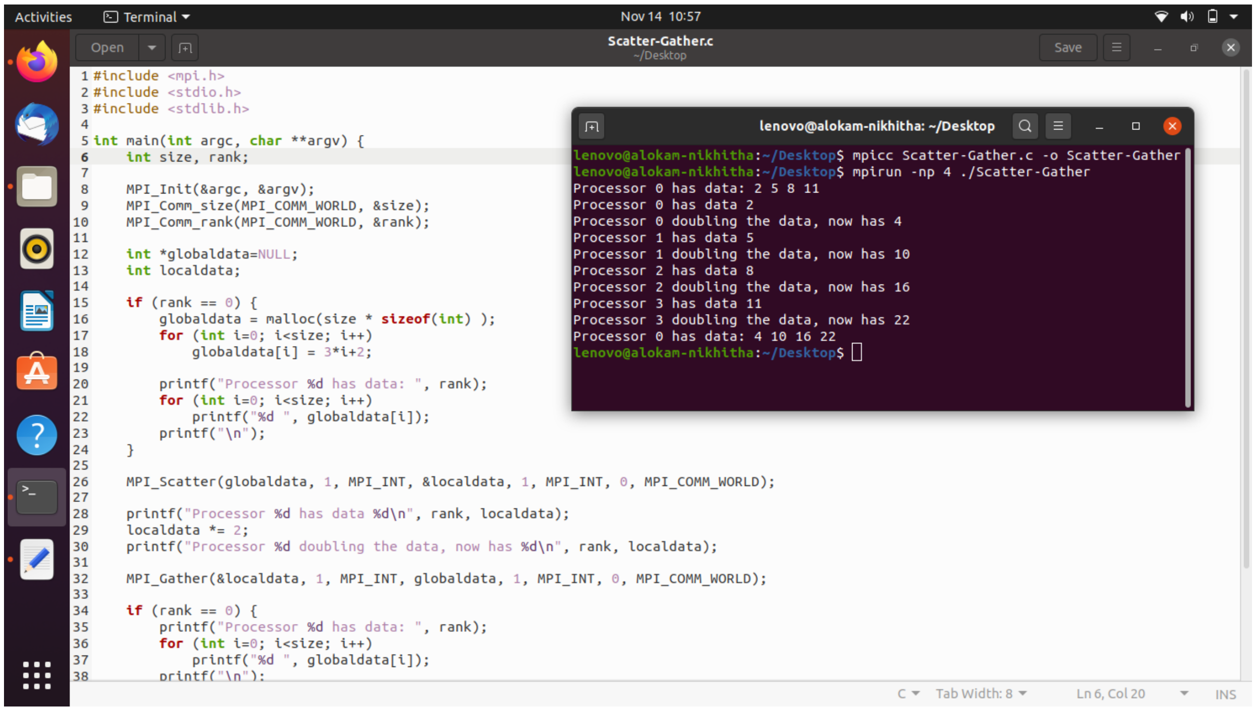
```c

    if (rank == 0) {
        printf("Processor %d has data: ", rank);
        for (int i=0; i<size; i++)
            printf("%d ", globaldata[i]);
        printf("\n");
    }

    if (rank == 0)
        free(globaldata);

    MPI_Finalize();
    return 0;
}
```

OUTPUT:



```
lenovo@alokam-nikhitha:~/Desktop$ mpicc Scatter-Gather.c -o Scatter-Gather
lenovo@alokam-nikhitha:~/Desktop$ mpirun -np 4 ./Scatter-Gather
Processor 0 has data: 2 5 8 11
Processor 0 has data 2
Processor 0 doubling the data, now has 4
Processor 1 has data 5
Processor 1 doubling the data, now has 10
Processor 2 has data 8
Processor 2 doubling the data, now has 16
Processor 3 has data 11
Processor 3 doubling the data, now has 22
Processor 0 has data: 4 10 16 22
lenovo@alokam-nikhitha:~/Desktop$
```

OUTPUT WITH CODE:



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

int main(int argc, char **argv) {
    int size, rank;

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

    int *globaldata=NULL;
    int localdata;

    if (rank == 0) {
        globaldata = malloc(size * sizeof(int) );
        for (int i=0; i<size; i++)
            globaldata[i] = 3*i+2;

        printf("Processor %d has data: ", rank);
        for (int i=0; i<size; i++)
            printf("%d ", globaldata[i]);
        printf("\n");
    }

    MPI_Scatter(globaldata, 1, MPI_INT, &localdata, 1, MPI_INT, 0, MPI_COMM_WORLD);

    printf("Processor %d has data %d\n", rank, localdata);
    localdata *= 2;
    printf("Processor %d doubling the data, now has %d\n", rank, localdata);

    MPI_Gather(&localdata, 1, MPI_INT, globaldata, 1, MPI_INT, 0, MPI_COMM_WORLD);

    if (rank == 0) {
        printf("Processor %d has data: ", rank);
        for (int i=0; i<size; i++)
            printf("%d ", globaldata[i]);
        printf("\n");
```

```
lenovo@alokam-nikhitha:~/Desktop$ mpicc Scatter-Gather.c -o Scatter-Gather
lenovo@alokam-nikhitha:~/Desktop$ mpirun -np 4 ./Scatter-Gather
Processor 0 has data: 2 5 8 11
Processor 0 has data 2
Processor 0 doubling the data, now has 4
Processor 1 has data 5
Processor 1 doubling the data, now has 10
Processor 2 has data 8
Processor 2 doubling the data, now has 16
Processor 3 has data 11
Processor 3 doubling the data, now has 22
Processor 0 has data: 4 10 16 22
lenovo@alokam-nikhitha:~/Desktop$
```