ASSIGNMENT NO-3

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

   int main(int argc, char* argv[]) {
   int rank, size;
   int num[20]; // N=20, n=4

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

   for (int i = 0; i < 20; i++)
      num[i] = i + 1;

   if (rank == 0) {
      int s[4];
      printf("Distribution at rank %d \n", rank);

      for (int i = 1; i < 4; i++)
         MPI_Send(&num[i * 5], 5, MPI_INT, i, 1, MPI_COMM_WORLD); // N/n i.e. 20/4=5

      int sum = 0, local_sum = 0;
      for (int i = 0; i < 5; i++) {
         local_sum = local_sum + num[i];
      }
      for (int i = 1; i < 4; i++) {
      MPI_Recv(&s[i], 1, MPI_INT, i, 1, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
      }
    printf("local sum at rank %d is %d\n", rank, local_sum);
```

```c
        sum = local_sum;

        for (int i = 1; i < 4; i++)

            sum = sum + s[i];

        printf("final sum = %d\n\n", sum);

    } else {

        int k[5];

        MPI_Recv(k, 5, MPI_INT, 0, 1, MPI_COMM_WORLD, MPI_STATUS_IGNORE);

        int local_sum = 0;

        for (int i = 0; i < 5; i++) {

            local_sum = local_sum + k[i];

        }

        printf("local sum at rank %d is %d\n", rank, local_sum);

        MPI_Send(&local_sum, 1, MPI_INT, 0, 1, MPI_COMM_WORLD);

    }

    MPI_Finalize();

    return 0;

}
```
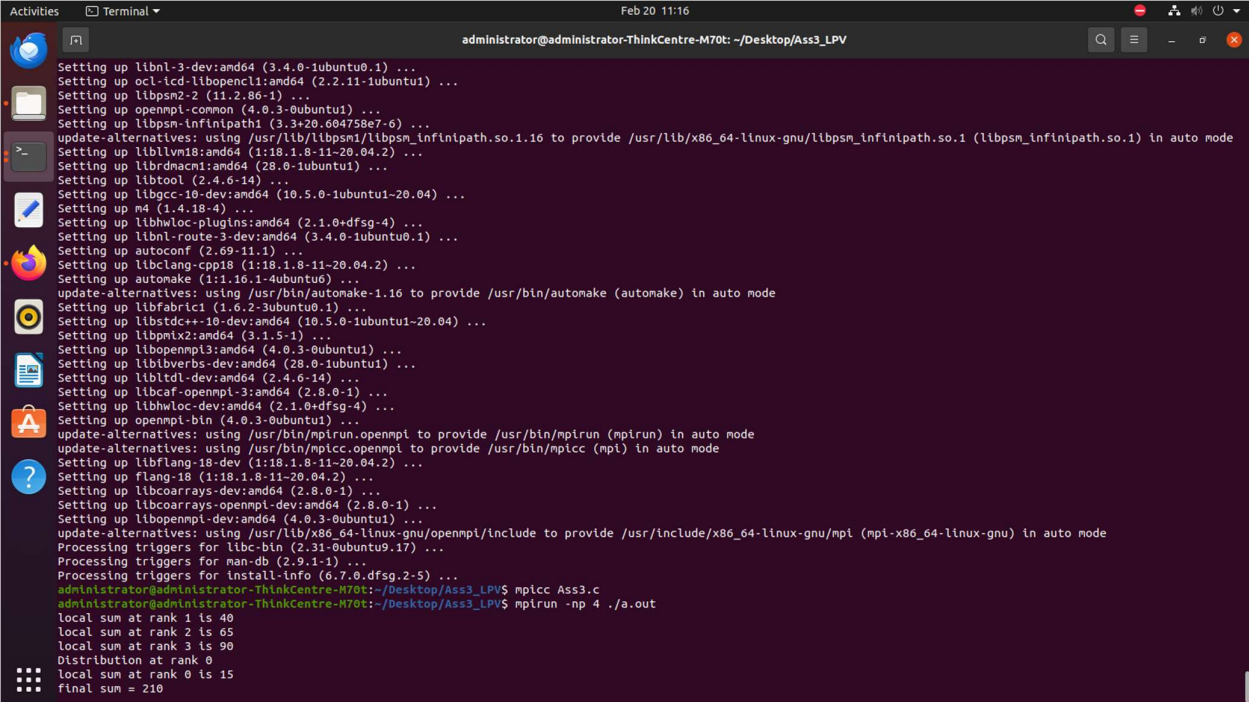
Output