Assessed Workshop Task

5. The file “WarAndPeace.txt” on Canvas contains the entire text of the book “War and Peace” by Leo Tolstoy. Write an MPI program to count the number of times each letter of the alphabet occurs in the book. Count both the upper case and the lowercase as the same. Ignore any letter with accents such as “é” and so on.

Your MPI program should work with any number of processes from 1 to 100. Only Process rank 0 (zero) should read in the file and send the **appropriate** chunk of file to each other process. The other processes should not read in the file.

You should submit this program as “workshoptask1.c” as part of your final portfolio submission. You can also upload it to the formative submission point for formative feedback.

Code:

```
#include <stdio.h>

#include <mpi.h>

#include <ctype.h>

#include <stdlib.h>


int main(int argc, char **argv)

{

  int size, rank, count = 0;

  char *data;

  MPI_Init(NULL, NULL);

  MPI_Comm_size(MPI_COMM_WORLD, &size);

  MPI_Comm_rank(MPI_COMM_WORLD, &rank);

  char filename[] = "WarAndPeace.txt";


  if (rank == 0)

  {

    FILE *fptr = fopen(filename, "r");

    if (fptr == NULL)

    {

      printf("Error Reading the file");

      exit(-1);

    }
```

```
    while (fgetc(fptr) != EOF)
    {
      count++;
    }
    fseek(fptr, 0, SEEK_SET);
    data = (char *)malloc(count * sizeof(char));
    int start = 0;
    char c;
while ((c = fgetc(fptr)) != EOF)
    {
      data[start++] = c;
    }
    fclose(fptr);
  }

  int local[26] = {0};
  int global[26] = {0};
  int localCount = 0;

  if (rank == 0)
  {
    int chunks = count / size;
    int remainder = count % size;
    for (int i = 1; i < size; i++)
    {
      int start = i * chunks;
      int send_size = chunks;
      MPI_Send(&data[start], send_size, MPI_CHAR, i, 0, MPI_COMM_WORLD);
    }
    localCount = chunks + remainder;
    for (int i = 0; i < localCount; i++)
```

```
    {
      char alpha = tolower(data[i]);
      if (alpha >= 'a' && alpha <= 'z')
      {
        local[alpha - 'a']++;
      }
    }
 }
  else
  {
    MPI_Status status;
    MPI_Probe(0, 0, MPI_COMM_WORLD, &status);
    MPI_Get_count(&status, MPI_CHAR, &localCount);
    char *localData = (char *)malloc(localCount * sizeof(char));
    MPI_Recv(localData, localCount, MPI_CHAR, 0, 0, MPI_COMM_WORLD, &status);
    for (int i = 0; i < localCount; i++)
    {
      char alpha = tolower(localData[i]);
      if (alpha >= 'a' && alpha <= 'z')
      {
        local[alpha - 'a']++;
      }
    }
  }

  MPI_Reduce(local, global, 26, MPI_INT, MPI_SUM, 0, MPI_COMM_WORLD);

  if (rank == 0)
  {
    for (int i = 0; i < 26; i++)
    {
```

```
        printf("%c count = %d times \n", 'a' + i, global[i]);
      }
    }

    MPI_Finalize();
    return 0;
}
```

Output:

```
root@DESKTOP-1377DP0:~/DistributedSystem# mpicc workshoptask1.c -o workshoptask1
root@DESKTOP-1377DP0:~/DistributedSystem# mpiexec -n 4 ./workshoptask1
a count = 201333 times
b count = 34369 times
c count = 60655 times
d count = 117759 times
e count = 311362 times
f count = 54513 times
g count = 50909 times
h count = 166531 times
i count = 170616 times
j count = 2485 times
k count = 20282 times
l count = 96042 times
m count = 61286 times
n count = 183138 times
o count = 188699 times
p count = 44717 times
q count = 2320 times
r count = 146891 times
s count = 162138 times
t count = 224517 times
u count = 63884 times
v count = 26789 times
w count = 58937 times
x count = 4032 times
y count = 45906 times
z count = 2386 times
```