```
#include "Communication.h"

Histogram* referenzToRealData(Histogram** data, int size_data, int total_size) {
  if (total_size < size_data) {
    total_size = size_data;
  }

  Histogram *new_memory = (Histogram*) malloc (total_size * sizeof(Histogram));
  int i;
  for (i = 0; i < size_data; i++) {
    memcpy(new_memory+i, data[i], sizeof(Histogram));
    //new_memory[i].cursor = (*data[i]).cursor;
    //new_memory[]
  }


  return new_memory;
}

/** Sende die Histogramme an einen anderen Prozess.
 * @param node Prozess an den die Histogramme gesendet werden sollen.
 * @param data Ein Pointer auf die Histogramme die gesendet werden sollen.
 * @param size Die Anzahl der Histogramme die gesendet werden sollen.
 */
void sendHistogram(int node, Histogram **data, int size, MPI_Datatype *HISTOGRAM
_TYPE) {

    // kopiere die Inhalte hinter den referenzen in einen neuen Speicherbereich.
    Histogram *new_memory = referenzToRealData(data, size, size);

    // Sende die Anzahl der Histogramme
    MPI_Send (&size,1,MPI_INT,node,0,MPI_COMM_WORLD);

  MPI_Send(new_memory, size, *HISTOGRAM_TYPE, node, 0, MPI_COMM_WORLD);
  free(new_memory);

}

/**
 * EmpfÃ¤ngt die Histogramme von einem anderen Prozess.
 * @param node Der Prozess von dem die Histogramme empfangen werden.
 * @param size Referenz auf die Anzahl der Elemente.
 */
Histogram* receiveHistogram(int node, unsigned int *size_received, Histogram *da
ta, unsigned int size_data, MPI_Datatype *HISTOGRAM_TYPE, Histogram** ref_data_s
orted) {
    MPI_Status status;

    // Empfange die Anzahl der Histogramme
    MPI_Recv (size_received,1,MPI_INT,node,0,MPI_COMM_WORLD,&status);

    unsigned int max_size = size_data + (*size_received);

    // statt auf den unsortierten Daten ein realloc zu machen machen wir ein mal
loc und kopieren die Daten hinter den REferenzen dort hinein
    // um anschlieÃM-^_end an das ende die empfangenen Daten zu kopieren
    Histogram *new_memory = referenzToRealData(ref_data_sorted, size_data, max_s
ize);
    free(data); // Data wird jetzt frei gegeben
    data = NULL;

  MPI_Recv(new_memory+size_data, *size_received, *HISTOGRAM_TYPE, node, 0, MPI_C
```



```
OMM_WORLD, &status);

  return new_memory;

}

void sendSortedHistogram(int node, int ranks, Histogram* data, int size, MPI_Dat
atype *HISTOGRAM_TYPE) {
  int size_p = size / ranks;

  // Sende die Anzahl der Histogramme
    MPI_Send (&size_p,1,MPI_INT,node,0,MPI_COMM_WORLD);

  // node = 2
  // ranks = 4
  // size = 100;
  // size_p = 100 / 4 = 25
  // data+(3*25) = 75


  MPI_Send(data+(node*size_p), size_p, *HISTOGRAM_TYPE, node, 0, MPI_COMM_WORLD)
;
}

Histogram* receiveSortedHistogram(int node, unsigned int *size_received, MPI_Dat
atype *HISTOGRAM_TYPE) {
  MPI_Status status;

    // Empfange die Anzahl der Histogramme
    MPI_Recv (size_received,1,MPI_INT,node,0,MPI_COMM_WORLD,&status);

    Histogram *sorted = (Histogram*) malloc (sizeof(Histogram) * (*size_received
));

    MPI_Recv(sorted, *size_received, *HISTOGRAM_TYPE, node, 0, MPI_COMM_WORLD, &
status);

    return sorted;
}
```