This homework is the extension of homework #4.

1. Read the original IDs in each bucket from an input file and stored them in `original_bucket[m]`, where `m` is the number of original buckets and `original_bucket[m]` are array of the following user-defined type:
   ```c
   typedef struct {
       int old_index;
       int size;
       int *id;
   } bucket_type;
   bucket_type original_bucket[m]
   ```

2. Assume the original buckets of numbers are already stored in array `original_bucket[m][n]`, where there are `m` buckets with maximum size = `n`. As stated in homework 4, all the original buckets must be sorted in the decreasing order of bucket sizes (by using function `qsort()` provided by C library) before applying new mapping algorithm to store the numbers in the original buckets into the `new_buckets[M][N]` (declared as int `new_buckets[M][N]`) where `N` can be set to be equal or larger than `n` and `M` is smaller than `m`.

3. Modify your homework 4 so that your program can answer:
   A. which original bucket is stored in which sorted bucket before compression,
   B. which original bucket is stored in which new bucket after compression,
   C. given the new bucket `i`, how many original buckets are contained in new bucket `i` and what are the indices of these original buckets contained in new bucket `i`.
   D. Compute the compression ratio(N) defined as ratio of the number of original IDs before and after compression, where \( N \geq n \) and draw a curve for compression ratio(N) with \( N = n \) to \( 3n \) to show how compression ratio varies.