CSI 2131 File Management — Winter 2005
Computer Science — University of Ottawa

Homework Assignment #3
Indexing using Hashed File

(individual or in a group of 2, 100 points, weight 10%)
Group declaration due: Monday March 14 (in class)
Due:

This program will use the given solution for Asssignment#2 as a startup code.
In this program, you will substitute the Simple Index (Array) implemented in assignment#2 by a Hashed Index (File). The majority of the changes are in the `Index class`.

Hashing specification:

- Hashed file specifications
  Bucket size: B (number of keys to be stored in bucket)
  Number of Addresses: N
- Hashing Functions:
  $h1(k, N) = (k^2 \text{ div } 1048576) \bmod N$; $h2(k, N) = (k \text{ div } 1280) \bmod N$;
  $g1(k, N) = (k^2 \text{ div } 65536) \bmod N$; $g2(k, N) = (k \text{ div } 65536) \bmod N$.
- Collision Resolution Methods:
  Progressive Overflow using $h1$ or $g1$.
  Double Hashing using primarily $h1$ or $g1$, and secondarily $h2$ or $g2$.

The following aspects (among others) of the **Index class** should be changed:

- Constructor:
  The constructor method receives as parameters:
  - an opened `fstream` where the hashed index file is to be stored,
  - the bucket size `B`,
  - the number of hash addresses `N`,
  - a pointer to hash function 1
  - a pointer to hash function 2 (this is an optional parameter, which is given if using double hashing, but it is omitted if using progressive overflow).

  The constructor must prepare the file for future insertions, by creating the file consisting of all empty buckets.

- INSERT:
The Insert method is modified in order to insert an item into the hashed file using the specified collision resolution method. Every time an insertion is made, the average search length, maximum search length and packing density are updated (these are new variables stored in this class).

- SEARCH:
The Search method is modified in order to search for an item in the hashed file using the specified collision resolution method.

- AVERAGESEARCHLENGTH, MAXSEARCHLENGTH, PACKINGDENSITY
New methods that return these quantities, which have been stored in this class.

Your main program still receives the datafile name and operations file name from the command line. However, it must now test several variations of the hashed index and create the corresponding file, as specified in the following table:

Hash file name	hash functions	N	B	Method
hf1.txt	$h1$	1009	1	progressive overflow
hf2.txt	$h1, h2$	1009	1	double hashing
hf3.txt	$h1$	419	2	progressive overflow
hf4.txt	$h1, h2$	419	2	double hashing
hf5.txt	$g1$	1511	1	progressive overflow
hf6.txt	$g1, g2$	1511	1	double hashing
hf7.txt	$g1$	277	3	progressive overflow
hf8.txt	$g1, g2$	277	3	double hashing

For each of the above files, your program must create the index for the datafile, and process the given operations, similarly to the previous assignment. For each of these files, store into a table, the main statistics (average search length, maximum search length and packing density) after all operations have been performed. After all the files have been processed, the program must report a summary table, which includes for each entry on the above table (in the given order), its main statistics.

Documentation and Style: Please, follow the same guidelines as in previous assignments.
What and how to submit your assignment: Please, read the course's policy on plagiarism and collaboration. By submitting this assignment you are automatically acknowledging understanding of the policy. Create a directory/folder named `a3` containing all the program files (.h and .cpp) needed to create your project. EVERY FILE MUST CONTAIN A HEADER WITH THE FOLLOWING INFO FOR EACH STUDENT IN THE PAIR: Student Name, Student Number, Course and Section. One of the students in the group zips the folder and submits the zipped file as the group's assignment#3 submission to webCT. Only one student in the group submits the program files; the other student submits a simple text file containing the name and student number of the students in the group.