0.1. Students

Name	Email	ONID
Bea McDonnell	mcdonneb@oregonstate.edu	933642077
Tristan 'Tryss' Goucher	gouchetr@oregonstate.edu	934650327

0.2. Instructions

Assume that we have a relation **Employee(id, name, bio, manager-id)**. The values of **id** and **manager-id** are integers each with the fixed sizes of 8 bytes. The values of **name** and **bio** are character strings and take at most 200 and 500 bytes, respectively. *Note that as opposed to the values of id and manager-id, the sizes of the values of name and bio are not fixed and are between 1 to 200 (500) bytes*. **The size of each page is 4096 bytes (4KB)**. The size of each record is less than the size of a page. Using the provided skeleton code with this assignment, write a C++ program that creates a hash index file for relation Employee using attribute id. Your program must also enable users to search the created index by providing the id of a record.

- **The Input File:** The input relation is stored in a CSV file, i.e., each tuple is in a separate line and the fields of each record are separated by commas. Your program must assume that the input CSV file is in the current working directory, i.e., the one from which your program is running, and its name is **Employee.csv**. We have included an input CSV file with this assignment as a sample test case for your program. Your program must correctly create hash indexes and search for id(s) using them for other CSV files with the same fields as the sample file.
- **Index Page Creation:** Your program must first read the input **Employee** relation and build a *basic hash index for the relation using attribute id*. Your program must store the hash index in a file, **EmployeeIndex.dat** (Binary file like the previous assignment) on the current working directory. **Your index file must be a binary data file rather than text / csv**. You must use one of the methods explained in our lectures on storage management for storing variable-length records and the method described for storing pages of variable-length records to store records and pages in the new data file. They are also explained in Sections 9.7.2 and 9.6.2 of the Cow Book, respectively. If your submitted program does not use these formats and page data structures to store data in the data file, it does not get any points.
- **Index File and Page Structure:** In each page, records are written from top to bottom, and the overflow page index is written at the end. If there's no overflow page, you should write –1. The slot directory is written to the end of the page before the overflow page index in reverse order. This arrangement allows you to follow the i-th record's offset and length in the i-th entry of the slot directory from the bottom of the page. Indexes and their positions are tracked using pageDirectory, and positions are assigned to the pages using nextFreePosition. You must use the hash function, h = id mod 2^8.
- **Searching the Data File:** After creating the hash-indexed file, your program must accept a list of Employee id values in its command line and search the file for all records with the given *id(s)*. As you have stored the data using page structures and indexed those using Hash-indexing, you must compute the hash value and go to the