The assignment should be done in groups of two students. You must turn in the source code of your program through Canvas. Each group must submit only one file that contains the full name, OSU email, and ONID of every member of the group.

1: Storage Management (9 points)

The objective of this assignment is to learn how to store information on and implement data structures for external storage.

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 or C++ program that stores relation *Employee* in a single file on the external storage, i.e., hard disk, and accesses its records.

- **The Input File:** Your program must first read the input Employee relation and store it on a new file on disk. 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 create the new file and store and access records on the new file correctly for other CSV files with the same fields as the sample file.
- **Data File Creation and Its Page Structure:** Your program must store the records of the input CSV file in a new data file with the name *EmployeeRelation* in the current working directory. 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 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.
- **Searching the Data File:** After finishing the file creation, your program must accept an Employee *id* in its command line and search the file for all records with the given id. The user of your program should be able to search for records of multiple *ids*, one *id* at a time.
- **Main Memory Limitation:** During the file creation and search, your program must keep up to three pages in main memory at any time. The submitted solutions that use more main memory will *not* get any points.
- Each student has an account on hadoop-master.engr.oregonstate.edu server, which is a Linux machine. You must ensure that your program can be compiled and run on this machine. You can use the following bash command to connect to it:

```
> ssh your_onid_username@hadoop-master.engr.oregonstate.edu
```

Then it asks for your ONID password and probably one another question. To access this server, you must be on campus or connected to the Oregon State VPN.

- You can use the following commands to compile and run C++ code:

```
> g++ -std=c++11 main.cpp -o main.out

> main.out
```

- **Skeleton Code:** We have included the files that contain the skeleton code to generate the required *EmployeeRelation* file. You can make changes to these files adhering to the assignment requirements.
- **Grading Criteria:** The program must follow the guidelines on organizing records in pages and the limits on the number of pages in the main memory. It must also return correct answers for search queries. The programs that do not implement the record and page data structures according to the guidelines given in the lectures will not get any points. Partial implementations will get partial credits. The programs that do not follow memory limitation requirements do not get any points. The programs that implement the aforementioned restrictions but do not answer any query from our test set correctly will get only the minimum (1-2) points. The programs that answer some search queries correctly and fail on others will get partial credits.