

# Homework: Storage & Buffer Management (w4)

[New Attempt](#)

**Due** May 3 by 11:59pm    **Points** 10    **Submitting** a file upload

## Introduction

The objective of this assignment is to learn storing information and designing data structure on external memories.



## What you must do

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 size 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 stores relation **Employee** and access its records in a single file on the external storage, i.e., hard disk.

- Your program must first read an input **Employee** relation and store in a new file on disk. The input relation is stored in a CSV file, i.e., each tuple is in a separate line and 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.
- Your program must store the records of the input CSV file in a new file with the name **EmployeeRelation** on the current working directory. You may use one of the methods explained for storing variable-length records and the method described on storing pages (blocks) of variable-length records in our lectures on storage management to store records and pages. They are also explained in Sections 9.7.2 and 9.6.2 of Cow Book, respectively.

- During the file creation, your program may keep up to *three pages* (blocks) in main memory at any time. The submitted solutions that use more main memory will not get any points.
- After finishing the file creation, your program should accept an **Employee id** in its command line and search the file for all records of the given id. Like file creation, your program may use up to *three pages* in main memory at any time. The submitted solutions that use more main memory will not get any points for implementing this lookup operation. The user of your program may search for records of multiple ids, one id at a time.
- Each student has an account on **hadoop-master.engr.oregonstate.edu** server, which is a Linux machine. You should 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 connect to the Oregon State VPN.

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

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

```
> main.out
```


## Necessary files

---

Input file: **Employee.csv** (<https://canvas.oregonstate.edu/courses/1939345/files/99013991?wrap=1>)\_   
([https://canvas.oregonstate.edu/courses/1939345/files/99013991/download?download\\_frd=1](https://canvas.oregonstate.edu/courses/1939345/files/99013991/download?download_frd=1))

Following files contain the skeleton code to generate the required EmployeeRelation file. You may make changes to these files adhering to the assignment requirements.

**main.cpp** (<https://canvas.oregonstate.edu/courses/1939345/files/99014006?wrap=1>)\_   
([https://canvas.oregonstate.edu/courses/1939345/files/99014006/download?download\\_frd=1](https://canvas.oregonstate.edu/courses/1939345/files/99014006/download?download_frd=1))

(<https://canvas.oregonstate.edu/courses/1939345/files/99014006?wrap=1>) **classes.h**  
(<https://canvas.oregonstate.edu/courses/1939345/files/99014014?wrap=1>)\_ 

([https://canvas.oregonstate.edu/courses/1939345/files/99014014/download?download\\_frd=1](https://canvas.oregonstate.edu/courses/1939345/files/99014014/download?download_frd=1))

## What to turn in

The assignment is to be turned in before Midnight (by 11:59pm) on May 3. You may turn in the source code of your program through Canvas. The assignment may be done in groups of two students. Each group may submit only one file that contains the full name, OSU email, and ONID of every member of the group.

## Grading criteria

The program must follow the guidelines on organizing records in pages and the limits on the number of pages in main memory. It must also return correct answers for search queries. The programs that do not implement the record and page data structures according the guidelines given in the lecture 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 minimum (1-2) points. The programs that answer some search queries correctly and fail on others will get partial credits.

### Storage & Buffer Management

Storage & Buffer Management						
Criteria	Ratings					Pts
Description of criterion	<b>10 pts</b> <b>Full Marks</b>	<b>9 pts</b> <b>Minor Error</b> Your program doesn't work properly if the ID is not in Employee.csv file.	<b>6 pts</b> <b>Partial Credit</b> Some queries were returned correctly.	<b>2 pts</b> <b>Major Error</b> Your program do not return any queries correctly	<b>0 pts</b> <b>No Marks</b> You did not meet the assignment requirements	10 pts
Total Points: 10						