The assignment is to be turned in before Midnight (by 11:59pm) on February 9th. You should turn in the solutions to writing part of this assignment (questions 1 and 2) as a PDF file through Canvas. The solutions should be produced using editing software programs, such as LaTeX or Word, otherwise they will not be graded. You should turn in the source code to each programming question (question 3) separately through Canvas. Thus, each group will have two distinct submissions in Canvas for this assignment. The assignment should be done in groups of two students. Each submission must contain the full name, OSU email, and ONID of every member of the group.

1: Tree Indexing (2 points)

Assume that our block size is 30 bytes. In this question, each node in a B+ tree must fit in a block and its size must be as close as possible to the size of a block. Answer the following parts using B+ trees.

(a) Assume that all data values are integers with a fixed size of 8 bytes and each record pointer takes at most 4 bytes. Find a B+ tree whose height changes from 2 to 3 when the value 20 is inserted. Note that the height of a tree is the depth of its deepest node plus one. For example, the height of a tree with a single node is 1. Show your structure before and after the insertion. (1 point)

(b) Assume that all data values are integers with a fixed size of 4 bytes and each record pointer takes at most 4 bytes. Find a B+ tree in which the deletion of the value 40 leads to a redistribution. Show your structure before and after the deletion. You may use a different B+ tree than the one you used in part (a). (1 point)

2: Indexing (1 point)

(a) Consider the following relational schema:
Emp(*eid*:integer, *ename*:string, *age*:integer, *salary*:real)

The underlined attribute is the key for this relations. Consider the following SQL query:

```
Select *
From Emp
Where age = 20 and salary > 20000
```

Describe a B+ tree index on Emp that improves the running time of this query more than every other index over most Emp relation instances. You should indicate the attributes, their order in the index, and whether the index should be clustered.

3: Sorting on external storage (7 points)

(a) Consider a file with records of the following structure:

```
Emp (eid (integer), ename (string), age (integer), salary (double))
```

Fields of types *integer*, *double*, and *string* occupy 4, 8, and 40 bytes, respectively. Assume that each (I/O) block can fit at most one record (tuple) of the input file. Implement the Two-Pass Multi-way Merge Sort algorithm for the file ***Emp.csv*** in C/C++ using the skeleton code posted with this assignment. The sorting should be based on the attribute ***eid***. There are at most 22 blocks available to the sort algorithm in main memory, i.e., the **algorithm cannot use more than 22 blocks**.

- 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.
- The result of the sort must be stored in a new CSV file. The file that stores the relation *Emp* is ***Emp.csv***.
- Your program must assume that the input file is in the current working directory, i.e., the one from which your program is running.
- The program must store the result in a new CSV file with the name **EmpSorted.csv** in the current working directory.
- Your program must run on hadoop-master. Each student has an account on *hadoop-master.engr.oregonstate.edu* server, which is a Linux machine. You should use this machine to test your program. You can use the following *bash* command to connect to it:

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

  You can access this server on campus. If you want to access the server from off-campus, you may need to use the VPN to access the campus network.
- You must name the file that contains the source code of the main() function **main3.cpp**.
- You may use following commands to compile and run C++ code:

  ```
  > g++ -std=c++11 main3.cpp -o main3.out
  > main3.out
  ```