The assignment is to be turned in before Midnight (by 11:59pm) on March 2nd. You should turn in the solutions to the written part of this assignment (questions 1 and 2) as a PDF file through Canvas. These 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 (questions 3 and 4) separately through Canvas. Thus, each group will have three distinct submissions in Canvas for this assignment. The assignment should be done in groups of two students.

1: Query processing Algorithms (1.5 points)

Consider the natural join of the relation *R(A,B)* and *S(A,C)* on attribute A. Neither relations have any indexes built on them. Assume that R and S have 80,000 and 20,000 blocks, respectively. The cost of a join is the number of its block I/Os accesses. If the algorithms need to sort the relations, they must use two-pass multi-way merge sort. You may choose the join algorithms in your answers from the ones taught in the class.

(a) Assume that there are 10 blocks available in the main memory. What is the fastest join algorithm for computing the join of R and S? What is the cost of this algorithm? (0.5 point)

(b) Assume that there are 350 blocks available in the main memory. What is the fastest join algorithm to compute the join of R and S? What is the cost of this algorithm? (0.5 point)

(c) Assume that there are 200 blocks available in the main memory. What is the fastest join algorithm to compute the join of R and S? What is the cost of this algorithm? (0.5 point)

2: Query processing (2 points)

(a) Assume that the entire of relation R(A,B) fits in the available main memory but relation S(A,C) is too large to fit in the main memory. Find a fast join algorithm, i.e., an algorithm with the lowest number of I/O access, for the natural join of R and S. Justify that your proposed algorithm is the fastest possible join algorithm to compute the natural join of R and S. Next, assume that there is a clustered index on attribute A of relation S. Explain whether or how this will change your answer. (1 point)

(b) Consider relations R(A,B) an S(A,C) that each have 1 million tuples and are too large to fit in main memory. A data scientist wants to compute 10000 (sample) tuples of the natural join of R and S very fast. Since it is too time-consuming to compute the full natural join of R and S, the data scientist selects 1% of relation R and 1% of relation S and computes their join. Explain whether this algorithm returns the desired results. If it does not, propose an efficient algorithm that returns the desired result without computing the full natural join of R and S. (1 point)

3: Sort-merge Join Algorithms (5.5 points)

(a) Consider the following relations:

```
Dept (did (integer), dname (string), budget (double), managerid (integer))
Emp (eid (integer), ename (string), age (integer), salary (double))
```

Fields of types *integer*, *double*, and *string* occupy 4, 8, and 40 bytes, respectively. Each block can fit at most one tuple of an input relation. There are at most 22 blocks available to the join algorithm in the main memory. Implement the optimized sort-merge join algorithm for $Dept \bowtie_{Dept.managerid=Emp.eid} Emp$ in C++.

- Each input relation is stored in a separate CSV file, i.e., each tuple is in a separate line and fields of each record are separated by commas.
- The result of the join must be stored in a new CSV file. The files that store relations Dept and Emp are Dept.csv and Emp.csv, respectively.
- Your program must assume that the input files are 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 join.csv in the current working directory.
- Your program must run on hadoop-master.engr.oregonstate.edu. Submissions should also include the g++ command (including arguments) that was used to compile the program. Each student has an account on the *hadoop-master.eecs.oregonstate.edu* server, which is a Linux machine. You can use the following *bash* command to connect to it:

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

  It will prompt you for your ONID password. You will need to be connected to the VPN in order to access the server.
- You must name the file that contains the source code of the main() function main3.cpp. If you place your source code in multiple files, you may submit all of them in a single zip file.
- You may use following commands to compile and run C++ code:

  ```
  > g++ main3.cpp -o main3.out
  > main3.out
  ```

4: External Memory Sorting (6 points)

(a) Consider the following relation:

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

Fields of types *integer*, *double*, and *string* occupy 4, 8, and 40 bytes, respectively. Each block can fit at most one tuple of an input relation. There are at most 22 blocks available to the sort algorithm in the main memory. Implement the multi-pass multi-way sorting for the relation *Emp* in C++. The relation should be sorted by *eid.*

- 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.engr.oregonstate.edu. Submissions should also include the g++ command (including arguments) that was used to compile the program. Each student has an account on *hadoop-master.eecs.oregonstate.edu* server, which is a Linux machine. You can use the following *bash* command to connect to it:

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

  It will prompt you for your ONID password. You will need to be connected to the VPN in order to access the server.
- You must name the file that contains the source code of the main() function main4.cpp. If you place your source code in multiple files, you may submit all of them in a single zip file.
- You may use following commands to compile and run C++ code:

  ```
  > g++ main4.cpp -o main4.out
  > main4.out
  ```