Homework 03

[2023-1] 데이터사이언스를 위한 컴퓨팅1(M3239.005500 001)

***Due: 2023*년 *4*월 *21*일 금요일 *23*시 *59*분**

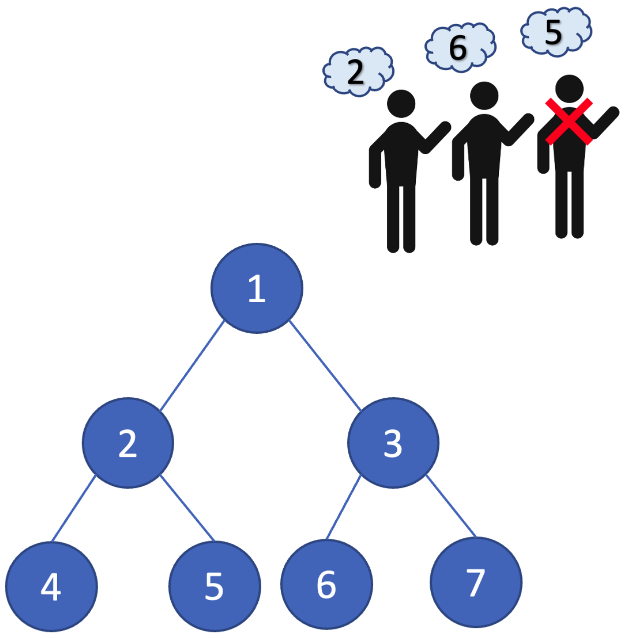
1. Seat Arrange Problem [40pts]

GSDS students are trying to rearrange the seats for the new semester. The seats numbers are as follows:

- The root seat is numbered 1.
- Saying that one seat has number K, the number of its left child is $2 \times K$, and its right child is $2 \times K + 1$

There has been an argument between GSDS students to occupy good seats, and Sarah, the head of GSDS student committee came up with a solution:

- Line up the students in a row.
- Let each student have the seat they want in order of the line.



In this case, the answer is 2 6 2

However, if a student encounters a seat occupied by another student who lined up first on the way to their desired seat, they cannot pass through it and cannot get the seat they want due to angry students who came first. The way to the desired seat includes the seat that the student wants to have; this means that pre-occupied seats cannot be taken by another student.

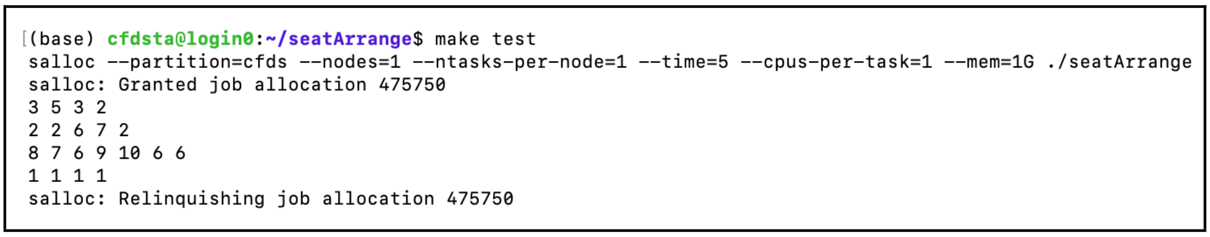
Given the solution proposed by Sarah, let's find out whether each student can have the seat they want. If so, ***print the seat they occupied*** and if not, please ***print the number of the first occupied seat encountered*** on the way to their desired seat. Assume that only valid inputs will be given (i.e., the seats desired by the students will always exist, and the number of students will match N).

```
int seatArrange(int N, int Q, vector<int>& queue)
   //N: number of seats in GSDS
   //Q: number of students
   //queue: desired seat for each student in a row
```

Implementation:

Do not modify `main` function in `seatArrange.cpp`. Feel free to declare any other functions.

`Makefile` is provided to help compilation and job scheduling. Please compile and run it at GSDS server. You can run test code by running "**make test**". Below is the results.



```
[(base) cfdsta@login0:~/seatArrange$ make test
 salloc --partition=cfds --nodes=1 --ntasks-per-node=1 --time=5 --cpus-per-task=1 --mem=1G ./seatArrange
 salloc: Granted job allocation 475750
 3 5 3 2
 2 2 6 7 2
 8 7 6 9 10 6 6
 1 1 1 1
 salloc: Relinquishing job allocation 475750
```

2. Simple Linear Regression [60pts]

Implement two classes: `DataFrame` and `Matrix`. `DataFrame` reads a `csv` file and can generate a `Matrix` object using specific columns. `Matrix` class stores a vector ($n \times 1$ matrix) or a matrix data and provide interface of matrix operations.

Instruction:

1. `DataFrame` class should have the following member functions:
   a. `int ReadData(std::string FileName, char sep, char comment, bool IsHeader)`

      `sep` is a separator of the file. Lines start with comment should be skipped. If `IsHeader` is true, the first line (after skipping) should be the header.
   b. `Matrix GetMatrix(int index[], int nColumn)`

Generate a `Matrix` using the columns in `DataFrame`. `index[]` is the array of the column index. Note that `0` means the first column.

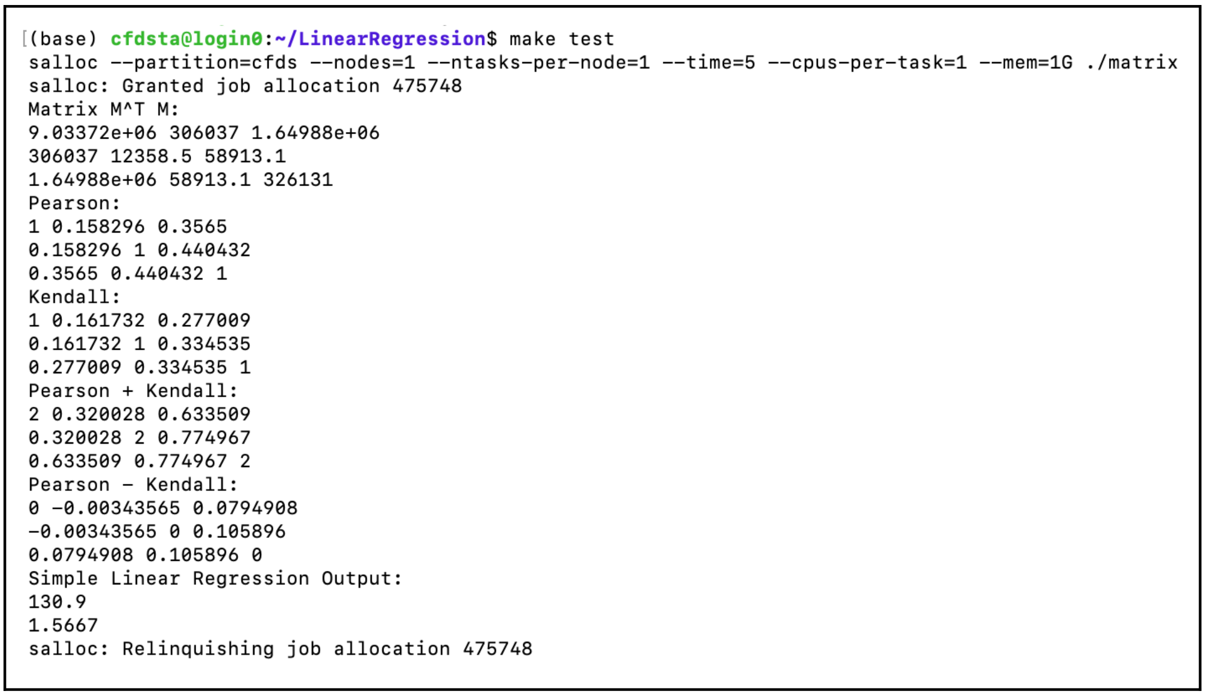
2. `Matrix` class should have the following member functions and operators:
   a. `+, -` : matrix addition and subtraction
   b. `*` : matrix multiplication
   c. `Matrix Transpose()` : return transpose matrix
   d. `Matrix GetSubVectorbyColumn(int column)` : return a specific column as a $n \times 1$ matrix.
   e. `void Print()` : print the values of the matrix
   f. `int GetNumRow()` : return the number of rows
   g. `int GetNumColumn()` : return the number of columns
   h. `double GetVal(int x, int y)` : return the value in `(x, y)` of the matrix. Note that `x` and `y` start from zero.

3. Implement the following function:
   a. `Matrix Cor(Matrix &mat, int method = 1)` : calculate the Pearson correlation (`method = 1`) or Kendall's tau (`method = 2`) of the all pairs of the columns. Returned matrix should be $m \times m$ matrix if the input matrix is $n \times m$. Diagonal elements of the returned matrix should be equal to 1. When calculating Kendall's tau, treat ties as concordant pairs. (i.e., treat (i, j) as a concordant pair when $x_i = x_j$ or $y_i = y_j$)
   b. `Matrix SimpleLinearRegression(Matrix &X, Matrix &Y)` : fit the simple linear regression of `X` and `Y`. `X` and `Y` should be $n \times 1$ matrices. Note that simple linear regression includes the intercept. So the returned value should be 2×1 Matrix of the coefficients.

Implementation:

Add all code related to class definition to `Matrix.h` and all implementation to `Matrix.cpp`. `main.cpp` have a test code.

`Makefile` is provided to help compilation and job scheduling. Please compile and run it at GSDS server. You can run test code by running "**make test**". You can check your result using data in coris.txt. Below is the results.



```
[(base) cfdsta@login0:~/LinearRegression$ make test
salloc --partition=cfds --nodes=1 --ntasks-per-node=1 --time=5 --cpus-per-task=1 --mem=1G ./matrix
salloc: Granted job allocation 475748
Matrix M^T M:
9.03372e+06 306037 1.64988e+06
306037 12358.5 58913.1
1.64988e+06 58913.1 326131
Pearson:
1 0.158296 0.3565
0.158296 1 0.440432
0.3565 0.440432 1
Kendall:
1 0.161732 0.277009
0.161732 1 0.334535
0.277009 0.334535 1
Pearson + Kendall:
2 0.320028 0.633509
0.320028 2 0.774967
0.633509 0.774967 2
Pearson - Kendall:
0 -0.00343565 0.0794908
-0.00343565 0 0.105896
0.0794908 0.105896 0
Simple Linear Regression Output:
130.9
1.5667
salloc: Relinquishing job allocation 475748
```

3. Submission Instruction

- Compress seatArrange directory (`seatArrange.cpp`) and LinearRegression directory (`Matrix.cpp, Matrix.h`) as a single file and report it to ETL.
- In seatArrange directory, **you cannot change `Makefile`.** You don't need to submit it.
- In LinearRegression directory, **you cannot change main.cpp and `Makefile`**. You don't need to submit them.
- The file name you submit should be **`YourAccount_HW03.zip`**. (ex: cfds123_HW03.zip). 파일명에 따라 채점된 점수가 자동으로 입력되는 시스템입니다. 파일명 준수 부탁드립니다. 해당 문제와 관련된 개인 과실로 0점 처리될 시 책임지지 않습니다.
- **Any disadvantages caused by not referring to the submission instructions will be scored without exception.**
- Make sure your code works well on the GSDS server. Your code will be scored automatically by the program on the GSDS server. If you don't follow the submission instruction, a penalty may occur.
- If you want to use your grace day, you must notify the TA by e-mail when submitting the homework. If you don't notify, we will judge that you want to save your grace day for the next homeworks, so your homework is considered unsubmitted. Even if you use your grace day, your homework should be submitted through ETL.