# DATABASE SYSTEMS (CSE 441)
# Assignment 2
**Deadline: Monday, 21st September 11:55 P.M.**

**Instruction:**
**1.** You are not allowed to use any external library/jar files in this assignment.
**2.** Plagiarism will not be tolerated. **Copy from any source and in any form (friends/seniors/internet) will fetch you straight 0 marks in all the assignments and quiz.**
**3**. Be careful about your submissions. You must strictly follow the upload format. Failure to do so will fetch you a zero.
**4.** Languages allowed to code are C/C++/Java.

**In this assignment you have to implement the two-phase merge sort algorithm to sort large number of records.**

**Algorithm Implementation:**
You need to consider following points:
**1.** The main memory to be used will be limited and specified by the input parameters.
For C/C++, use the dynamic memory allocation malloc and request 80% of the main memory specified in MB from the underlying OS and use the same memory for sorting. You shouldn't use array or global variable to allocate memory for the sorting variable.
For Java, use the argument -Xmx to restrict the heap size equal to the memory specified in MB.
2. The metadata file will contain the information about the size of the different columns (in bytes).
3. All the columns in the input file will be separated by a space.
4. The datatype for the column will be string only.
5. The number of columns can ranges from 1 to 100.
6. You program should be capable of sorting in both ascending and descending order (based on argument given).
7. Your program should run for different values of main memory allowed and different size of files ranging from MBs to GB s.
If we found your program is using main memory more than it is specified, you will lose marks.

**Input Format:**

**metadata.txt (**considered in the same folder as your executable file) containing the information about the column size.
<column name 1>,<size of the column>
<column name 2>,<size of the column>
<column name 3>,<size of the column>

.....
<column name n>,<size of the column>

**input.txt**

Containing the records with the column values separated by the space.
All the values will be in the string only.

**Command Line Inputs**

1. Input file name (containing the raw records)
2. Ouput file name (containing sorted records)
3. Main memory size (in MB)
4. Order code (asc/desc) asc means to sort in ascending order and desc means to sort in descending order
5. Columname 1
6. Columnname2
7. .....

**Example**

- ./sort input.txt output.txt 50 asc c0 c1
- ./sort input.txt output.txt 100 desc c0
- ./sort input.txt output.txt 100 asc c1 c0
- ./sort input.txt output.txt 500 asc c2

As in the first example, the file input.txt to be sorted in ascending order with 50 MB space based on column c0 and if any column have same value for c0 then on the basis of c1, available. (Similar to ORDER BY clause in SQL).
In case if the value is same for all the available columns, then the order should be the same as in the input file.

**Graph Generation:**

You have to generate two graphs:

**1)** FileSize(X-axis) v/s Time(Y-axis)
Take the main memory size as 100 MB and run you alogrithm for file size 5MB, 50MB, 500MB, 1GB, 2GB, 3GB, 4GB and 5GB and note the time taken for each run. Plot the graph.

**2)** Memory Size (X-axis) v/s Time (Y-axis)
Take a file of size 512MB and run your code with main memory argument as 10 MB, 25 MB, 100 MB, 250 MB, 512 MB and note the time taken for each run..
You can use any language or tool to plot the graph. (Matlab, Ms Excel).
You will need to provide the matrix and the image file the graph.

**Submission:**
Create a folder with the name **rollno_Assign2** and put the following into it:
**a)** Your source code in the folder named as **code**. (It should not contain the code used for graph generation)
**b)** A pdf file with the name **analysis.pdf** containing following information
    **i)** Configuration of the System
    **ii)** Both matrix used to plot the graph (in tabular format)
    **iii)** Both the image files of the graph
**c)** A bash file with the name **rollno.sh** that take to compile your code and run your code.
Compress the folder and upload the **rollno_Assign2.tar.gz**
You should not upload only any class files or the executable file.

**Error Handling:**
Check feasibility condition for two-way merge sort for provided memory constraints. If not satisfied, provide error message and halt. In this case, the output.txt should not generated.

**Test Data:**

To generate input file, visit http://www.ordinal.com/gensort.html and download **gensort** code to generate data.

The following command generates the first 100 tuples of three columns.
$> ./gensort -a 100 input.txt

To validate your algorithm you can use the **valsort** available on the same link.