# File Management and Processing:

# Assignment #2

## 100 points

Cairo University, Faculty of Computers and Information

**Delivery Instructions:**

1. Cheaters will be graded by ***–ve points, Don't copy any code from anywhere.***
2. Due Date: **Week starting January 1, 2021**
3. Team = max 3 students.
4. No late submission will be accepted.
5. Name your assignment with this format **CS215-YourGroup-TA-Student1ID-Student2ID-Student3ID-Assignment1.zip**
6. The assignment weighs 10 grades.
7.

Write a C++ program that stores sorted fixed-length **blocks of records** on disk as described in the lecture. Each Record stores exactly 2 integers as follows:

```
int iKey;

int iVal;
```

***m*** blocks are stored in a binary file. Each block contains ***n*** fixed length records, where:

- The first block in the binary file contains exactly 1 Record, where this record stores the index of the first non-empty block in "iKey" (iKey = -1 if all are empty) [in other words, stores where the linked list of blocks starts] and the index of the first empty block in "iVal" (iVal = -1 if all have records).

- If a block is empty, then the first Record in a block would have "iKey = -1" and iVal would store the index to next empty block.

- If a block is not empty, then the **FIRST Record in each block stores** :

    - The index to the next block in the linked list of blocks is stored in "iKey"-
    - The largest iKey stored in the block is stored in "iVal."

- Aside from the first Record in a block, the records in a block are sorted by values of "iKey".
- When a block has less than int (n/2) records, ALWAYS combine/merge with the previous block.

To illustrate how the linked list of blocks of records works, consider a block set containing 4 blocks, with each block containing 5 records. When initially empty:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| -1 | -1 | | | | | -1 | | | | | -1 | | | | |
| 1 | 2 | | | | | 3 | | | | | -1 | | | | |

# File Management and Processing:

## Assignment #2

### 100 points

Cairo University, Faculty of
Computers and Information

After we insert (1, 5), (4, 18), (5, 27), (2, 88) in this order:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | -1 | 1 | 2 | 4 | 5 | -1 | | | | | -1 | | | | |
| 2 | 5 | 5 | 88 | 18 | 27 | 3 | | | | | -1 | | | | |

After we insert (3, 20), (8, 4), (9, 11), (7, 15) in this order:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 1 | 2 | 3 | | 3 | 4 | 5 | 7 | | -1 | 8 | 9 | | |
| -1 | 3 | 5 | 88 | 20 | | 7 | 18 | 27 | 15 | | 9 | 4 | 11 | | |

After we delete 7, 2, and 4 in this order:

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 3 | 1 | 3 | 5 | | -1 | | | | | -1 | 8 | 9 | | |
| 2 | 5 | 5 | 20 | 27 | | -1 | | | | | 9 | 4 | 11 | | |

You must use the following functions headers

```
// for all the prototypes the Records set is stored in a file called cIndexFile

Bool CreateRecordFile(char *cIndexFile, int m, int n); // returns true if success and false if
failure. m is the number of blocks in the file and n is the number of records in a block

int InsertVal(char *cIndexFile, int iToken, int iKey); // returns index of block in which iToken
and iKey were stored and -1 if failed, where iKey is the key of the record, and iToken = iVal in
the record.

int GetKey(char *cIndexFile, int iBlock, int iRecord); // get value iKey stored in a given block
iBlock and given record iRecord – returns -1 if record on block is empty

int GetVal(char *cIndexFile, int iBlock, int iRecord); // get value iVal stored in a given block
iBlock and given record iRecord – returns -1 if record on block is empty

int GetBlockIndex (char *cIndexFile, int iToken); // get index of block containing iKey = iToken
and -1 if record does not exist

int GetRecordIndex (char *cIndexFile, int iToken); // get index of record containing iKey = iToken
and -1 if record does not exist

void DeleteKey (char *cIndexFile, int iToken); // delete record containing value iKey
= iToken

int FirstEmptyBlock(char *cIndexFile); // return the index of the first empty block.
```

Note: If there is an overflow in the block, don't distribute it; instead, split it.