Revision on CS215: File Organization and Processing

- Study the methods for field and record organization.
- Study how do the functions: seekp, seekg, tellp, tellg work in a file.
- Study the vail list and placement strategies for reclaiming spaces.
- Study all theoretical part of indexes.
- Understand how to apply the six methods of hashing given a table of data. Look at lecture examples.
- Train yourself to compute the IO time taken to sort large file. (look at lecture example)

Consider we want to store the data of a set of students in a file; every student has the following attributes

```
o  char ID[5];
o  char Name[50];     //max len = 50
o  char Address[50]; //max len = 50
```

- The file structure is as follows:
  - The fields separated by a delimiter.
  - Every record is preceded by its length.

The requirements:

1. Build primary index using the student **ID**
2. Build secondary index using the student **name** (for simplicity don't use inverted list)

Implement the following functions, taking into consideration <u>the two indexes.</u>

3. **Void Add ()** //ask user to enter full student data and insert this data into file, for simplicity, always insert new records in the end of file "Append".
4. **Void Delete()**// in this function you should ask user to enter ID for student to be deleted, if the ID not exists, print student not found. No avail list is required.
5. **Void Update(),**in this function you should ask user to enter ID for student to be updated, if the ID not exists, print to screen "student not found". If the new size is bigger than the old one, mark the old record as deleted and insert the new one in the end of file.
6. **Void PrintStudentbyID()**// in this function you should ask user to enter ID for student to be printed, in this functions you have to search in the primary index using Binary Search.
7. **Void PrintStudentbyName()**// in this function you should ask user to enter ID for student to be printed, in this functions you have to search in the index using Binary Search.
8. **Void compactFile()** //this function is to cleanup deleted records from the file.
   You may use this technique => copy all undeleted records to ostringstream object, then write the ostringstream object to the file in ios::trunc mode.

Hints:

- All Indexes must be sorted ascending
- Bind secondary index with primary key, not with record address (bind-at-retrieval)
- To search in an index you must use binary search.
- **NO NEAD TO** implement secondary index using **inverted list.**
- **ignore avail list in the delete operation.**
- No need to use **out_of_date** flag to keep the index integrity

You may Start using this class:

```
class FileManagementSystem
{

	int nextP;//next free slot in primary index
	int nextS;//next free slot in secondary index

	struct student
	{
		char ID[5];
		char Name[50];
		char Address[50];
	};
	struct PIndexRecord
	{
		char PK[5];
		int offset;
	};
	struct SIndexRecord
	{
		Char Name[50];
		int offset;
	};

	const static int indexSize = 100;
	PIndexRecord P_index[indexSize];
	SIndexRecord S_index[indexSize];
public:
	FileManagementSystem()
	{
		int nextP =0;
		int nextS=0;

	}

	//write the rest of functions here ……

};
```