File Management and Processing - Assignment 2

Write a C++ program that create, store and manipulate a set of ***n*** fixed-length records as B-tree index on a **binary file**. Each record consists of *m* descendants (*m* records IDs and *m* references to the actual records on a data file) + 1 integer to indicate leaf/no leaf status (The first integer of each node, 0→ means a leaf node, 1 → a non-leaf node). The file at creation time (if number of records = 10 and m= 5) should contain the following:

-1	1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	2	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	3	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	4	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	5	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	6	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	7	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	8	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	9	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

The first node (Node with index ZERO) will always have the index of the first empty node (at its second integer) in order to add a new record. Node with index ZERO will only be used to tell the next free node and no data to be added into it. Each empty node will have the index of the next empty node (at its second integer), forming a linked list of empty nodes. The index of the root node will be always the first node to insert into which is node with index 1.

Your Program should handle the cases on insertion, deletion and Search as detailed in the example.

Consider the following Insertions:

Insert: 3, 12

Insert: 7, 24

Insert: 10, 48

Insert: 24, 60

Insert: 14, 72

	3		7		10		14		24

The index file should look as follow:

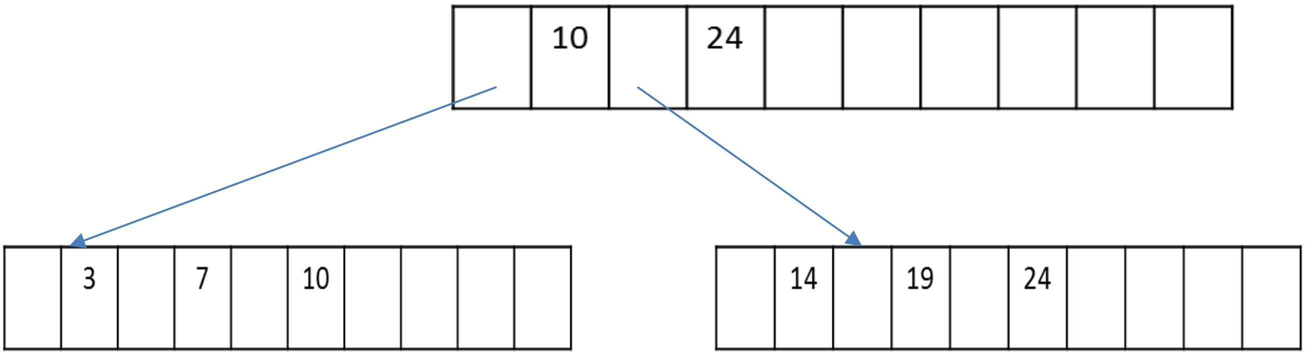
-1	2	-1	-1	-1	-1	-1	-1	-1	-1	-1
0	3	12	7	24	10	48	14	72	24	60
-1	3	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	4	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	5	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	6	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	7	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	8	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	9	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

After the previous insertions, Node 1 is a leaf node and all the references in it are pointing to records on data file.

File Management and Processing - Assignment 2

Consider the following Insertion:

Insert: 19, 84



The index file should look as follow:

-1	4	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	10	2	24	3	-1	-1	-1	-1	-1	-1
0	3	12	7	24	10	48	-1	-1	-1	-1
0	14	72	19	84	24	60	-1	-1	-1	-1
-1	5	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	6	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	7	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	8	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	9	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

After the previous insertions, Node 1 is a non-leaf node and all the references in it are pointing to other nodes on the index file.

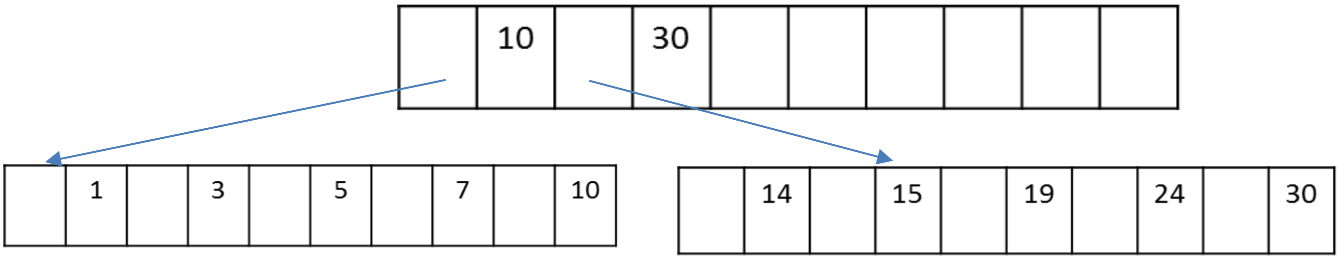
Consider the following Insertions:

Insert: 30, 96

Insert: 15, 108

Insert: 1, 120

Insert: 5, 132

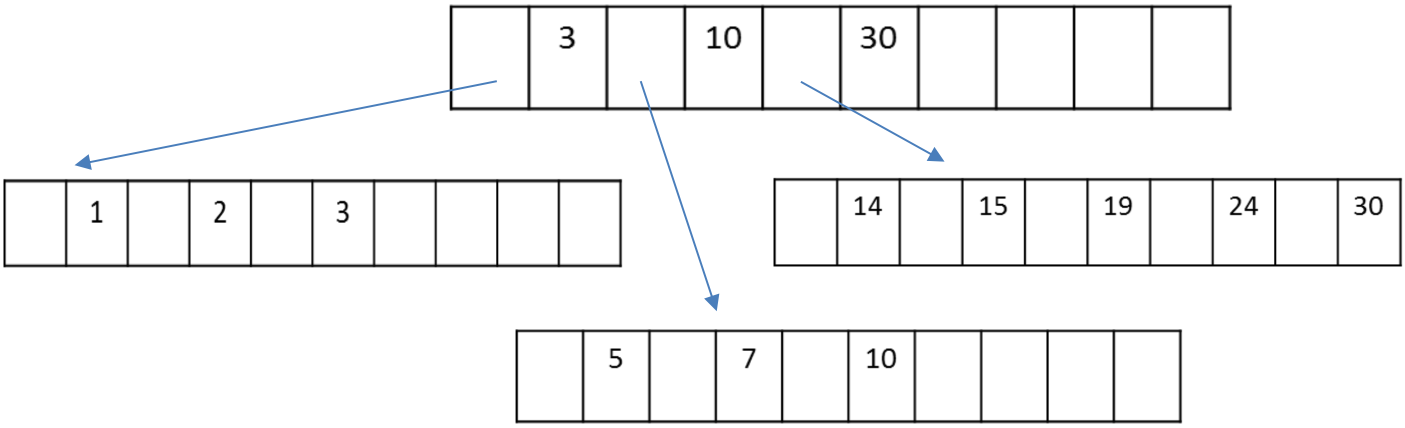


The index file should look as follow:

-1	4	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	10	2	30	3	-1	-1	-1	-1	-1	-1
0	1	120	3	12	5	132	7	24	10	48
0	14	72	15	108	19	84	24	60	30	196
-1	5	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	6	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	7	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	8	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	9	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

Consider the following Insertion:

Insert: 2, 144



The index file should look as follow:

-1	5	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	3	2	10	4	30	3	-1	-1	-1	-1
0	1	120	2	144	3	12	-1	-1	-1	-1
0	14	72	15	108	19	84	24	60	30	196
0	5	132	7	24	10	48	-1	-1	-1	-1
-1	6	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	7	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	8	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	9	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

File Management and Processing - Assignment 2

Consider the following insertions

Insert: 8, 156

Insert: 9, 168

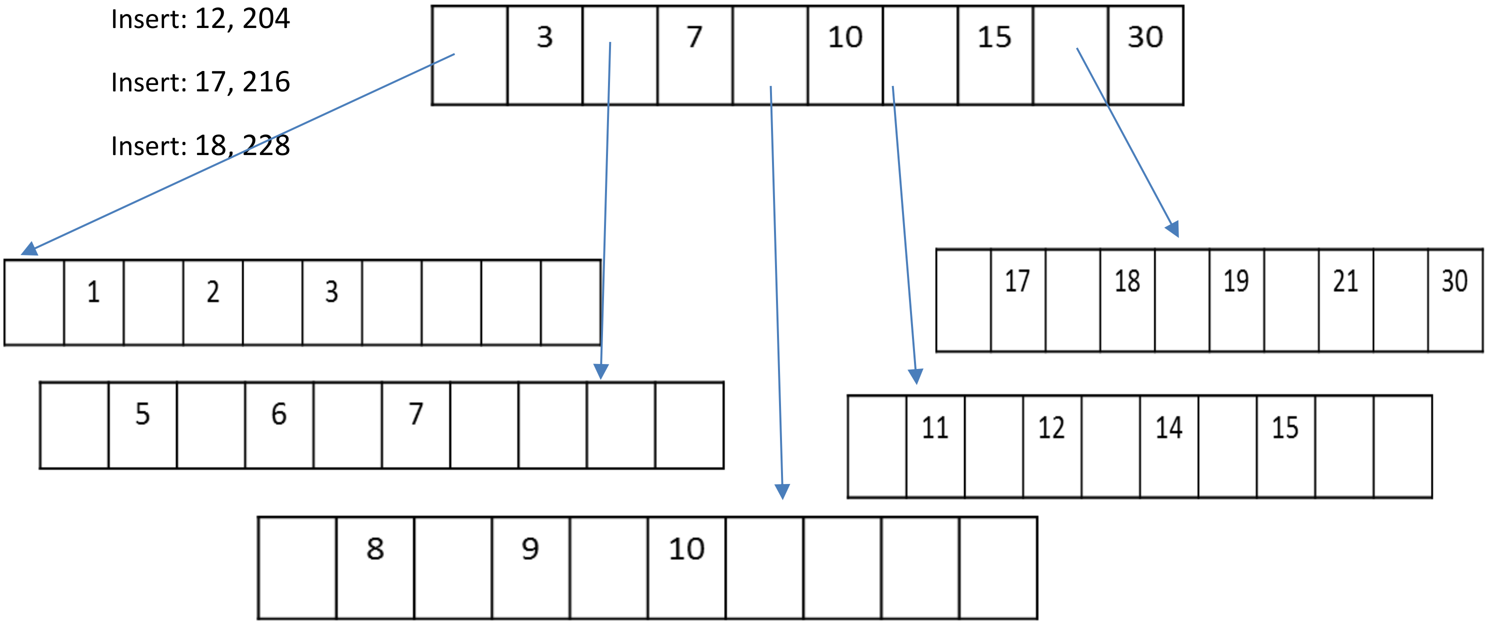
Insert: 6, 180

Insert: 11, 192

Insert: 12, 204

Insert: 17, 216

Insert: 18, 228

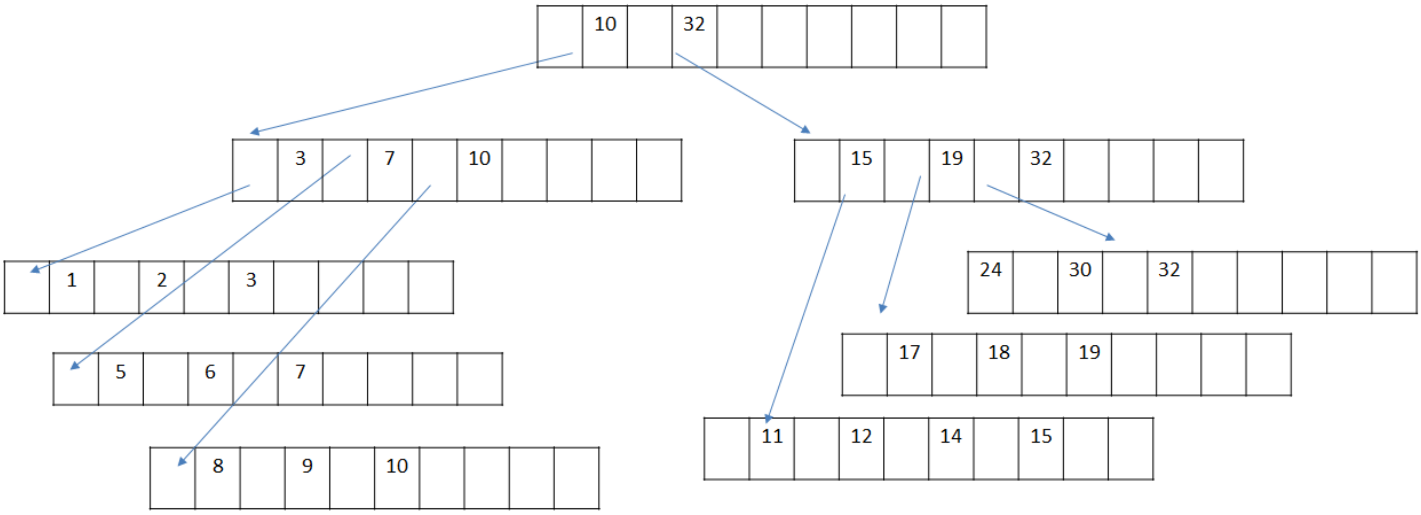


The index file should look as follow:

-1	7	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	3	2	7	4	10	5	15	3	30	6
0	1	120	2	144	3	12	-1	-1	-1	-1
0	11	192	14	72	12	204	15	108	-1	-1
0	5	132	6	180	7	24	-1	-1	-1	-1
0	8	156	9	168	10	48	-1	-1	-1	-1
0	17	216	18	228	19	84	24	60	30	196
-1	8	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	9	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

Consider the following insertion:

Insert: 32, 240

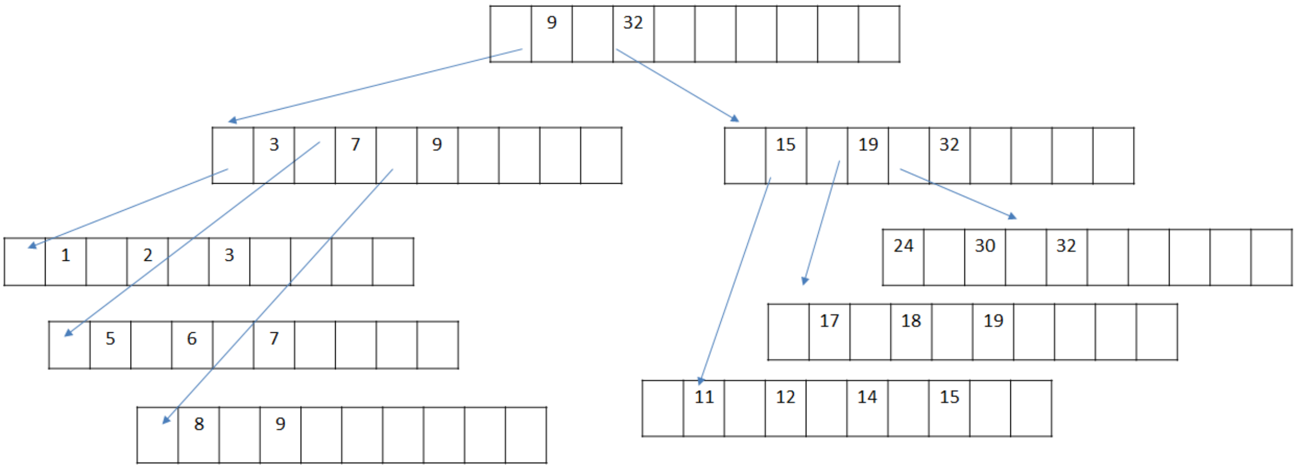


The index file should look as follow:

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	10	8	32	9	-1	-1	-1	-1	-1	-1
0	1	120	2	144	3	12	-1	-1	-1	-1
0	11	192	14	72	12	204	15	108	-1	-1
0	5	132	6	180	7	24	-1	-1	-1	-1
0	8	156	9	168	10	48	-1	-1	-1	-1
0	17	216	18	228	19	84	-1	-1	-1	-1
0	24	60	30	196	32	240	-1	-1	-1	-1
1	3	2	7	4	10	5	-1	-1	-1	-1
1	15	3	19	6	32	7	-1	-1	-1	-1

Consider the following deletion:

Delete: 10

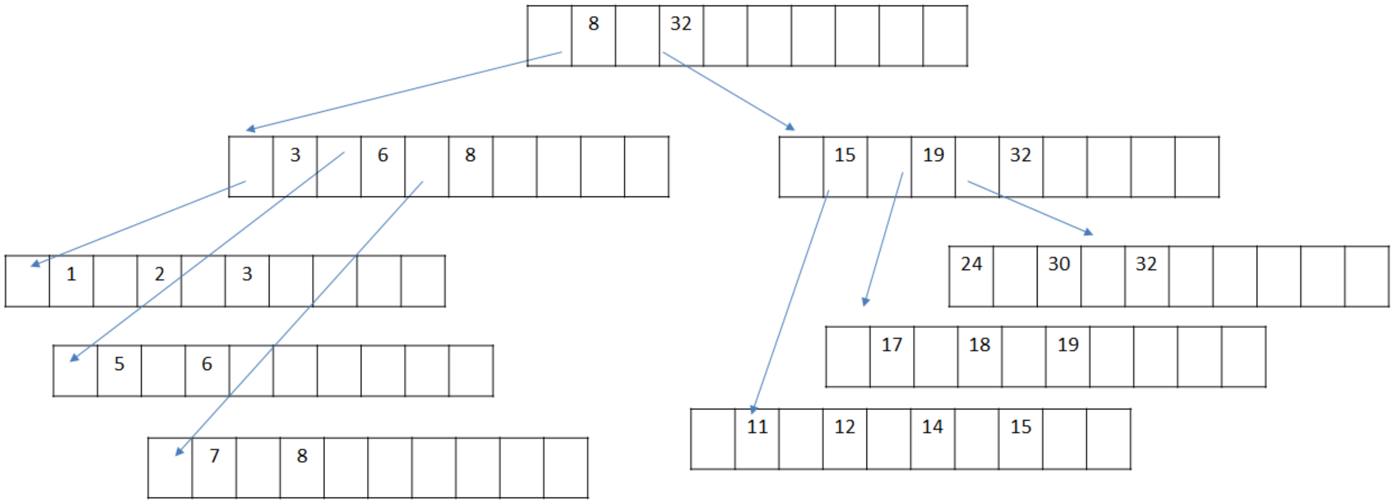


The index file would look as follow:

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	9	8	32	9	-1	-1	-1	-1	-1	-1
0	1	120	2	144	3	12	-1	-1	-1	-1
0	11	192	14	72	12	204	15	108	-1	-1
0	5	132	6	180	7	24	-1	-1	-1	-1
0	8	156	9	168	-1	-1	-1	-1	-1	-1
0	17	216	18	228	19	84	-1	-1	-1	-1
0	24	60	30	196	32	240	-1	-1	-1	-1
1	3	2	7	4	9	5	-1	-1	-1	-1
1	15	3	19	6	32	7	-1	-1	-1	-1

Consider the following deletion:

Delete: 9



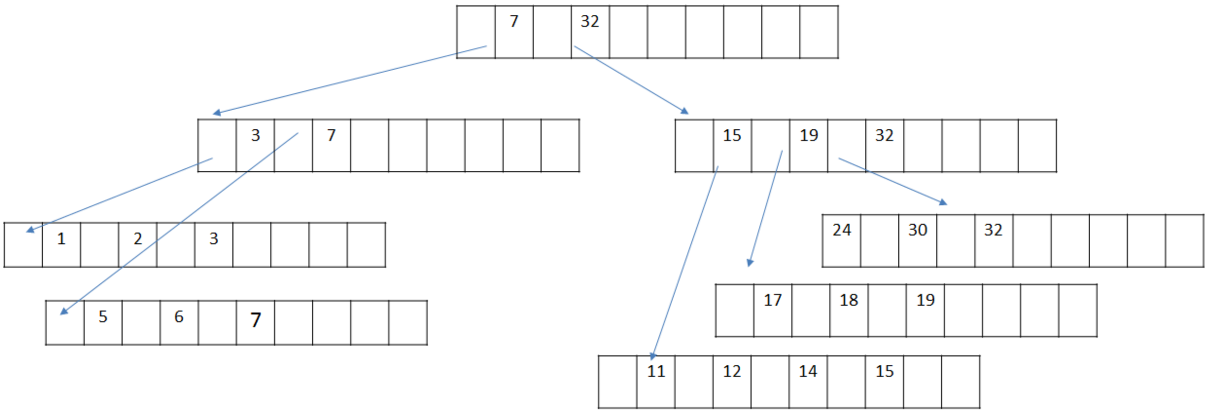
The index file would look as follow:

-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	8	8	32	9	-1	-1	-1	-1	-1	-1
0	1	120	2	144	3	12	-1	-1	-1	-1
0	11	192	14	72	12	204	15	108	-1	-1
0	5	132	6	180	-1	-1	-1	-1	-1	-1
0	7	24	8	156	-1	-1	-1	-1	-1	-1
0	17	216	18	228	19	84	-1	-1	-1	-1
0	24	60	30	196	32	240	-1	-1	-1	-1
1	3	2	6	4	8	5	-1	-1	-1	-1
1	15	3	19	6	32	7	-1	-1	-1	-1

File Management and Processing - Assignment 2

Consider the following deletions:

Delete: 8



-1	5	-1	-1	-1	-1	-1	-1	-1	-1	-1
1	7	8	32	9	-1	-1	-1	-1	-1	-1
0	1	120	2	144	3	12	-1	-1	-1	-1
0	11	192	14	72	12	204	15	108	-1	-1
0	5	132	6	180	7	24	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
0	17	216	18	228	19	84	-1	-1	-1	-1
0	24	60	30	196	32	240	-1	-1	-1	-1
1	3	2	7	4	-1	-1	-1	-1	-1	-1
1	15	3	19	6	32	7	-1	-1	-1	-1

In order to deliver your project, you **must** use the following functions header:

void CreateIndexFileFile (Char* filename, int numberOfRecords, int m) (0.5 Grade)

int InsertNewRecordAtIndex (Char* filename, int RecordID, int Reference) (3 Grades)

//insert function should return -1 if there is no place to insert the record or the index of the node where the new record is inserted if the record was inserted successfully.

void DeleteRecordFromIndex (Char* filename, int RecordID) (5 Grades)

void DisplayIndexFileContent (Char* filename) (0.5 Grade)

// this method should display content of the file, each node in a line.

int SearchARecord (Char* filename, int RecordID) (2 Grade)

// this method should return -1 if the record doesn't exist in the index or the reference value to the data file if the record exist on the index

Hint → at each insertion/deletion, keep an array on memory and store the sequence of nodes you visited (their indices) in order to get back to them later when updating those nodes

- **Project Delivery** will be week starting **31/12**
- Project should be done **On a Group of 4 students or less**. All the submitted code must be completely yours.
- Make sure to stick the functions headers as provided in the description.