Write a java program **(using java.io.RandomAccessFile class)** 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	2	3	4	5	6	7	8	9	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-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.

For 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

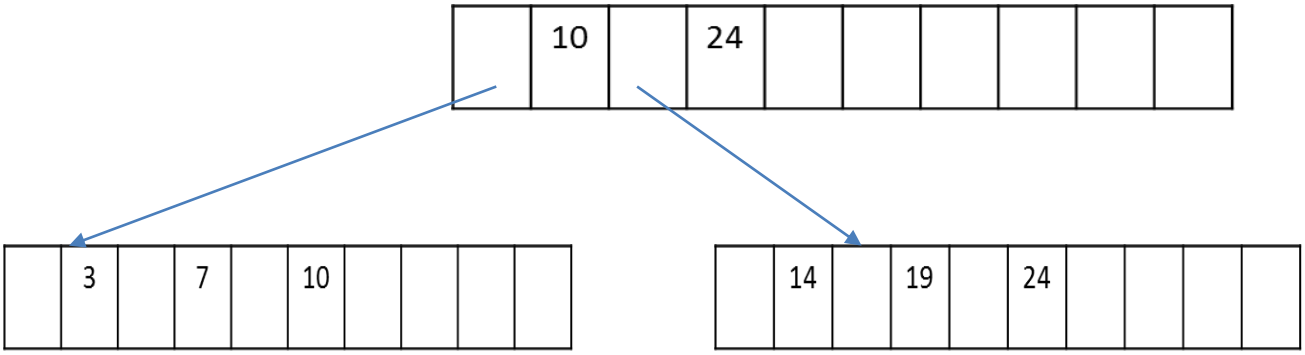
The index file should look as follow:

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

Consider the following Insertion:

Insert: 19, 84



The index file should look as follow:

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

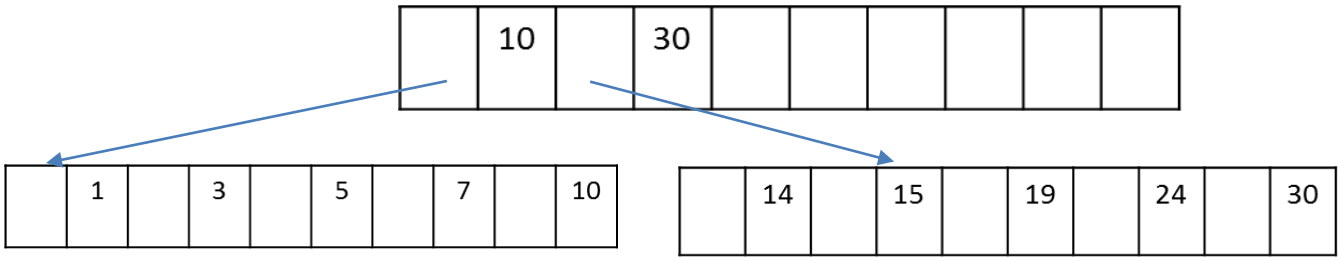
Consider the following Insertions:

Insert: 30, 96

Insert: 15, 108

Insert: 1, 120

Insert: 5, 132

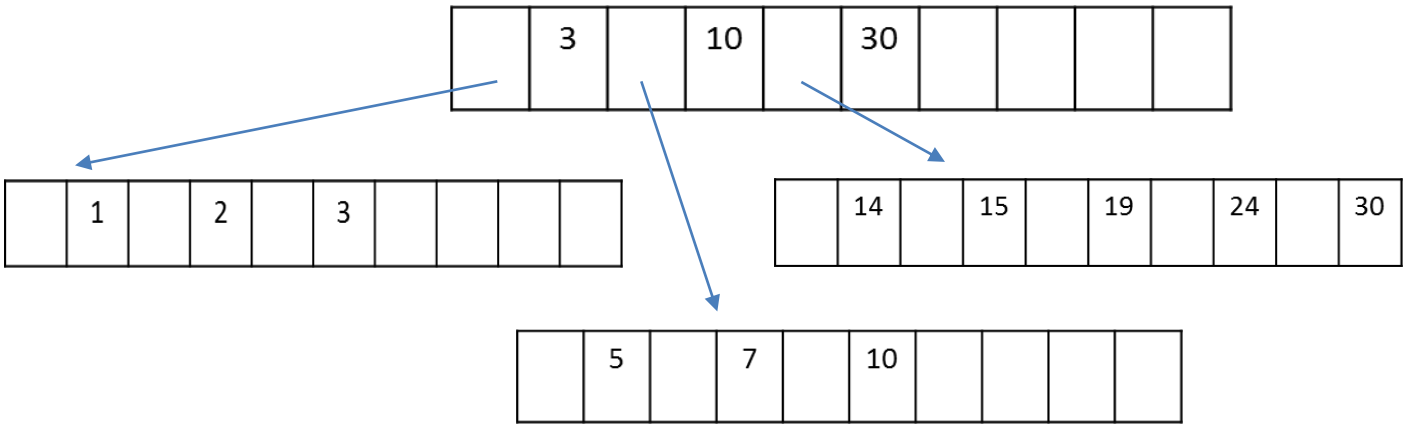


The index file should look as follow:

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

Consider the following Insertion:

Insert: 2, 144



The index file should look as follow:

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

Consider the following insertions

Insert: 8, 156

Insert: 9, 168

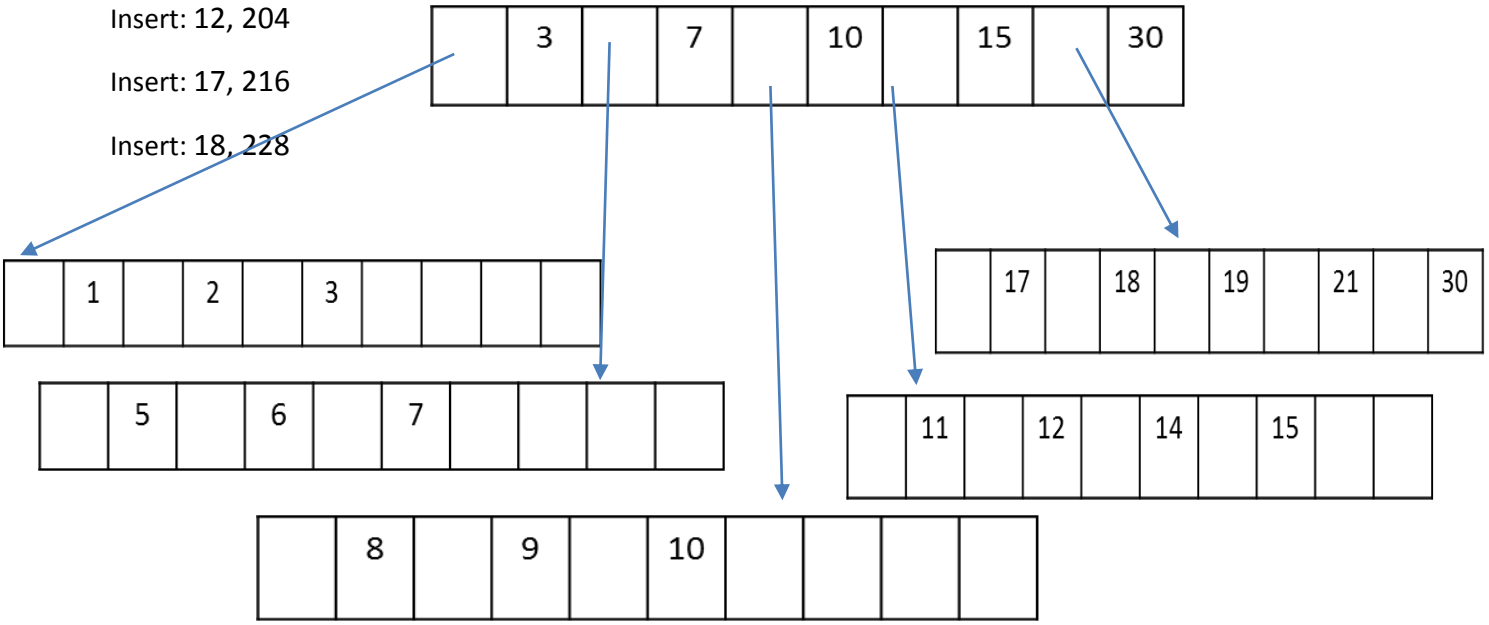
Insert: 6, 180

Insert: 11, 192

Insert: 12, 204

Insert: 17, 216

Insert: 18, 228

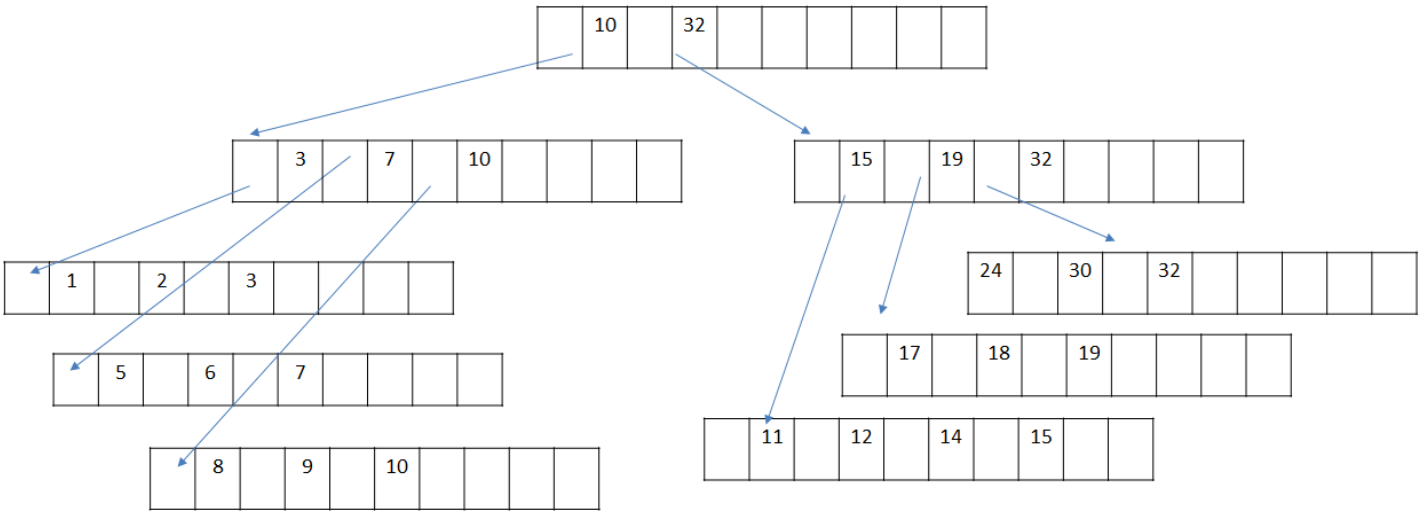


The index file should look as follow:

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

Consider the following insertion:

Insert: 32, 240

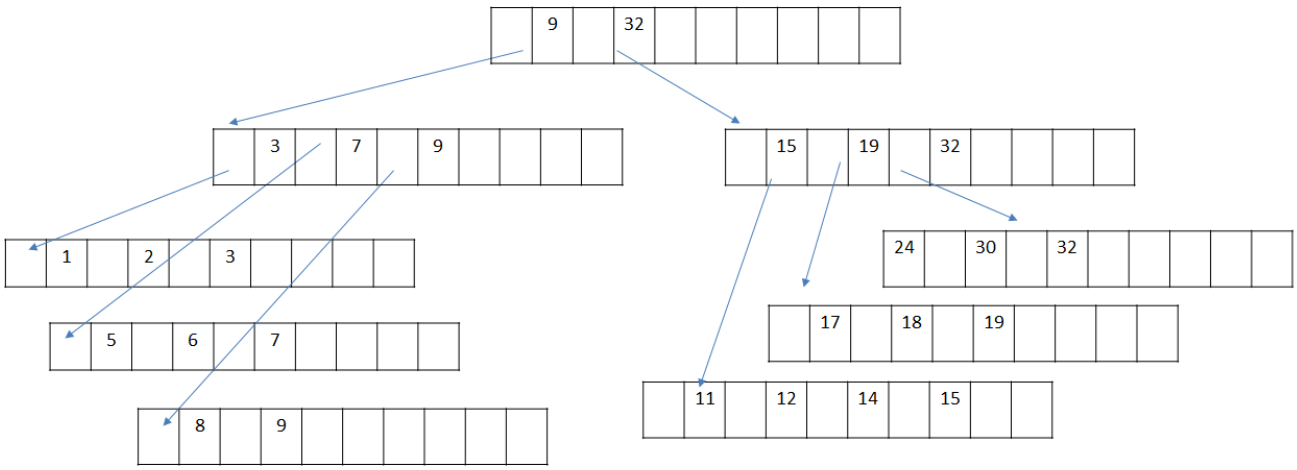


The index file should look as follow:

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

Consider the following deletion:

Delete: 10

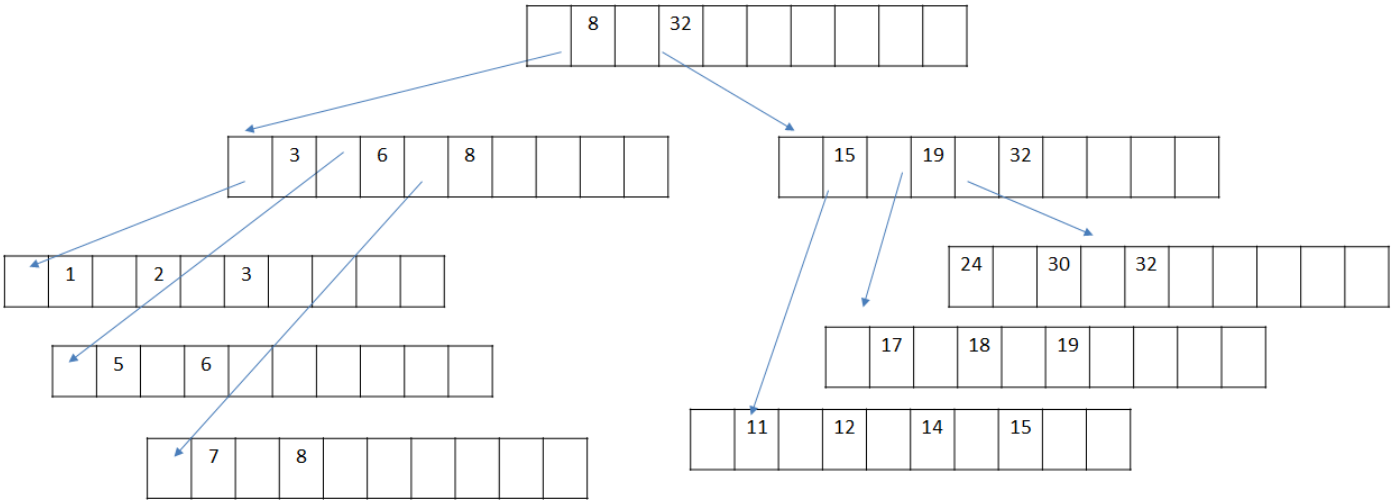


The index file would look as follow:

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

Consider the following deletion:

Delete: 9

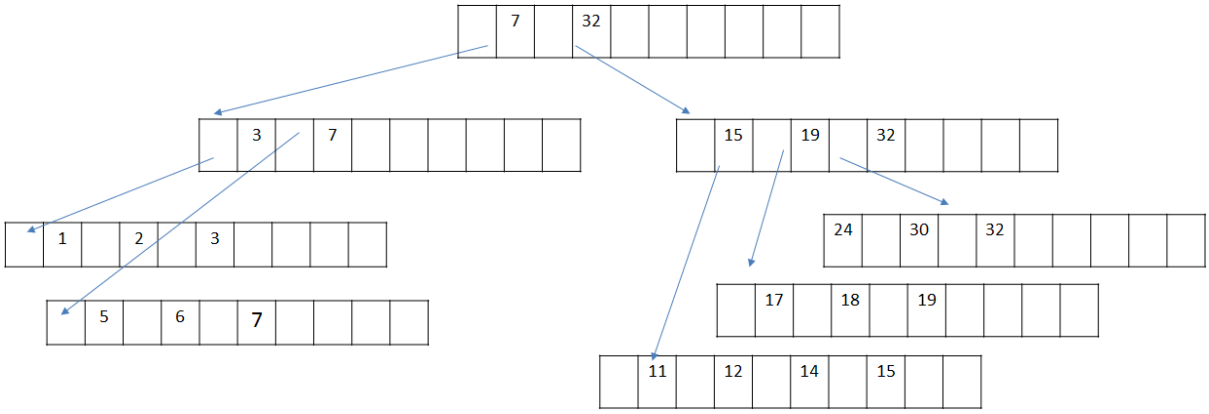


The index file would look as follow:

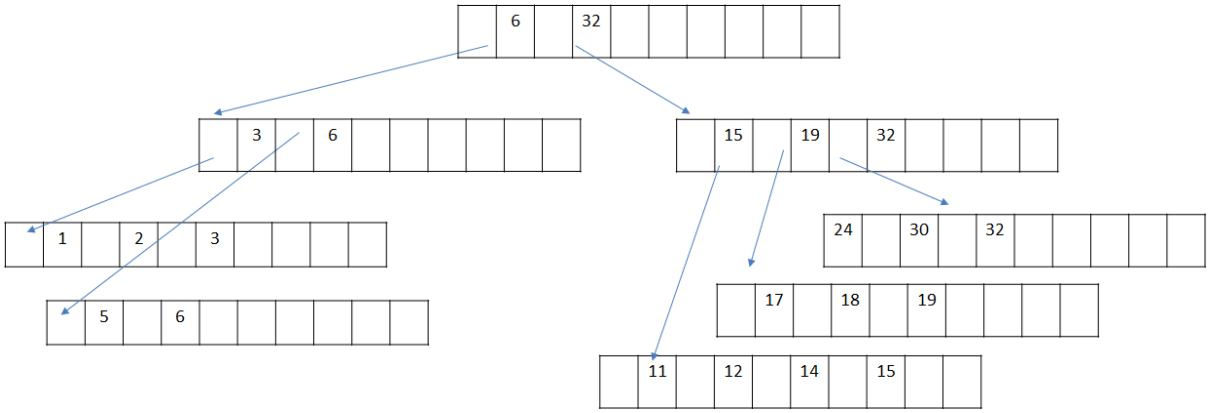
-1	1	0	0	0	0	0	0	1	1
-1	8	1	11	5	7	17	24	3	15
-1	8	120	192	132	24	216	60	2	3
-1	32	2	14	6	8	18	30	6	19
-1	9	144	72	180	156	228	196	4	6
-1	-1	3	12	-1	-1	19	32	8	32
-1	-1	12	204	-1	-1	84	240	5	7
-1	-1	-1	15	-1	-1	-1	-1	-1	-1
-1	-1	-1	108	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
-1	-1	-1	-1	-1	-1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

Consider the following deletions:

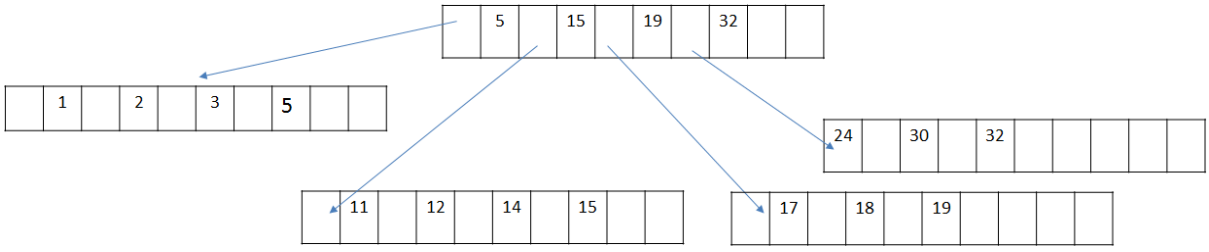
Delete: 8



Delete: 7



Delete: 6



The index file would look as follow:

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

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

void CreateIndexFileFile (String filename, int numberOfRecords, int m) (0.5 Grade)

int InsertNewRecordAtIndex String filename, int RecordID, int Reference) (4 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 (String filename, int RecordID) (7 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 On Thursday **19/5**

Project should be done **On a Group of 2 students or less**. All the submitted code must be completely yours. Your grade depends on delivering a running code and your understanding for the code.