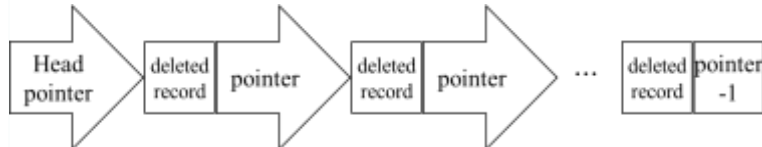
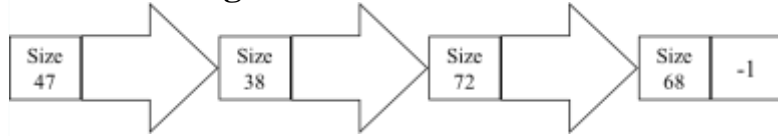


Second Internal Test

Sub:	File Structures						Code:	15IS62	
Date:	15/04/2019	Duration:	90 mins	Max Marks:	50	Sem:	VI	Branch:	ISE
Answer Any FIVE FULL Questions									

		Marks	OBE	
			CO	RBT
1 (a)	Define Indexing and its importance in File Structures. List and explain the different operations required maintain an indexed file	[10]	CO3	L1
ans	<p>Indexing is a structure containing a set of entries, each consisting of a key field and a reference field, which is used to locate records in a data file. It helps in faster access of records in a file if the size of the index file is small. Since index files are sorted on key field, binary search can be applied to find the presence of the key and use the reference field for performing a direct access to locate the record in single seek.</p> <p>Operations required to maintain an indexed file:</p> <ol style="list-style-type: none"> 1) Creating the data and index files. 2) Loading the index file to memory 3) Rewriting the index file from memory 4) Record addition 5) Record deletion 6) Record updating <p>Explain each operation in detail.</p>			
2 (a)	What is an Avail List? Explain how free spaces are reclaimed dynamically in fixed length and variable length record files.	[10]	CO2	L2
Ans	<p>Issues on reclaiming space quickly:</p> <ul style="list-style-type: none"> – How to know immediately if there are empty slots in the file? – How to jump to one of those slots, if they exist? <p>It is done by creating by linking all deleted records together using a linked list (Avail List).</p> <p>Fixed-Length Records</p>  <p>Variable-Length Records</p>  <p>Explain the mechanism of making use of avail list for reclaiming the space created by a deleted record.</p>			

3 (a) Write and explain Keysorting algorithm with an example. What are its limitations?

[10]

CO3	L2

Ans Keysorting: A sort performed by first sorting keys and then moving records.

- Read each record sequentially into memory, one by one.
- Save the key of the record and the location of the record, in an array (KEYNODES)
- After all records have been read, internally sort the KEYNODES array of record keys and locations
- Using the KEYNODES array, read each record back into memory a second time using direct access.
- Write each record sequentially into a sorted file.

```
int KeySort (FixedRecordFile & inFile, char * outFileName)
{
    RecType obj;
    KeyRRN * KEYNODES = new KeyRRN [inFile . NumRecs()];
    // read file and load Keys
    for (int i = 0; i < inFile . NumRecs(); i++)
    {
        inFile . ReadByRRN (obj, i); // read record i
        KEYNODES[i] = KeyRRN(obj.Key(), i); //put key and RRN into Keys
    }
    Sort (KEYNODES, inFile . NumRecs()); // sort Keys
    FixedRecordFile outFile; // file to hold records in key order
    outFile . Create (outFileName); // create a new file
    // write new file in key order
    for (int j = 0; j < inFile . NumRecs(); j++)
    {
        inFile . ReadByRRN (obj, KEYNODES[j].RRN); //read in key order
        outFile . Append (obj); // write in key order
    }
    return 1;
}
```

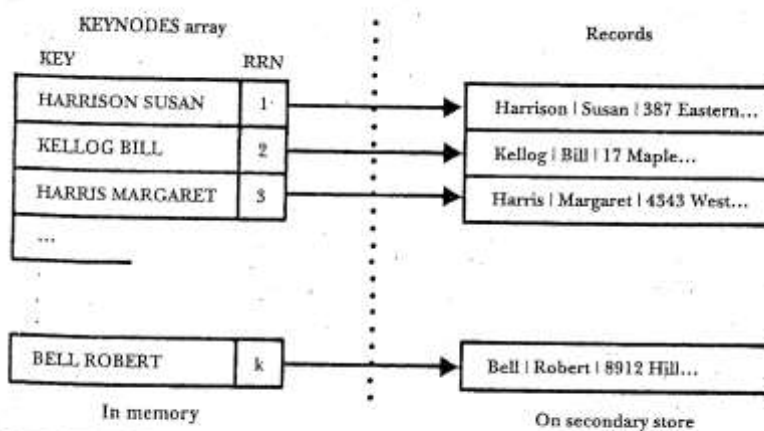


Figure 6.16 Conceptual view of KEYNODES array to be used in memory by internal sort routine and record array on secondary store.

Limitations:

- Only possible when the KEYNODES array is small enough to be held in memory.
- Each record must be read twice: Once sequentially and once directly.
- Each direct access requires a seek.
- Key sorting is a way to sort medium sized files.

4 (a) Explain the object oriented modeling for implementing consequential processing. Write an algorithm for consequential match using single loop, three way conditional statement. [10]

Ans Cosequential operations involve the coordinated processing of two or more sequential lists to produce a single output list. **Matching names in two lists: Matters to Consider:**

- **Initializing:** we need to arrange things so that the procedure gets going properly.
- **Getting and accessing the next list item:** we need simple methods to do so.
- **Synchronizing:** we have to make sure that the current item from one list is never so far ahead of the current item on the other that a match will be missed.
- **Handling end-of-file conditions:** Halt the program on reaching end of list1 or list2
- **Recognizing Errors:** Duplicate items or items out of sequence.

```

int Match (char * List1Name, char * List2Name,
          char * OutputListName)
{
    int MoreItems;// true if items remain in both of the lists

    // initialize input and output lists
    InitializeList (1, List1Name);// initialize List 1
    InitializeList (2, List2Name);// initialize List 2
    InitializeOutput (OutputListName);

    // get first item from both lists
    MoreItems = NextItemInList(1) && NextItemInList(2);

    while (MoreItems){// loop until no items in one of the lists
        if (Item(1) < Item(2))
            MoreItems = NextItemInList(1);
        else if (Item(1) == Item(2)) // Item1 == Item2
        {
            ProcessItem (1); // match found
            MoreItems = NextItemInList(1) && NextItemInList(2);
        }
        else // Item(1) > Item(2)
            MoreItems = NextItemInList(2);
    }
    FinishUp();
    return 1;
}

```

Figure 8.2 Cosequential match function based on a single loop.

Example showing matching of two sorted lists.

CO5	L2

5 (a) Construct a B-Tree for the following sequence of elements in the order of its insertion, illustrate state of the tree for every promotion and splitting:

C G J X N S U O A E B H I F

[10]

6 (a) Write an algorithm for Heap sort method for insertion. Show the construction of Heap for the following sequence: **F D C G H I B E A**

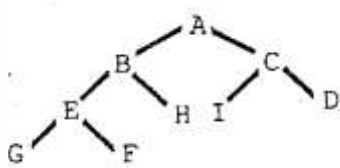
[10]

Ans Heap for the following sequence F D C G H I B E A.

Insert function for adding a new key to the heap:

```

Insert(NewKey) {
if (NumElements==MaxElements) return false;
NumElement++;
HeapArray[NumElements]= NewKey;
int k=NumElements;
int parent;
while (k>1){ // k has a parent
    parent=k/2;
    if (Compare(k, parent) >= 0)//already in order
        break;
    else
        Exchange(k, parent);
    k=parent;
}
return true;
}
    
```



CO3	L3
CO3	L3

7 (a) Construct an AVL tree for the following elements in the order of its insertion:
Q W E R T Y U I O P

[10]

CO3	L3
-----	----

8 (a) Write a short notes on:

1. Placement Strategy
2. Internal Sorting
3. Selective Index
4. Replacement Selection.

[10]

CO3	L2
-----	----

Ans **Pinned Records:** Placement strategy is used for selecting the right slot for reuse from the avail list when used with variable length records. Different strategies are 1. Best Fit, 2. Worst Fit, 3. First Fit.

Internal Sorting: If the entire contents of the file can be held in memory, then the entire file from the disk can be read into memory and then do the sorting there

Selective Index: Index on a subset of records, Provides a selective view of the data. Will contain only some part of the entire index data.

Replacement Selection: It increases the run length during merge sort by making use of multiple heaps. One is called the primary heap and the other as secondary heap.

--	--