BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI (RAJ.)

II Semester 2018-19 CS F211 Data Structures and Algorithms Online-1

Date: 06-Mar-2019 Duration:160 minutes Weightage: 54M

Note: Do not change the prototypes of given functions and names of global variables.

A company ABC Inc maintains the ID number of every employee as a 9 digit integer:

d1 d2 d3 d4 d5 d6 d7 d8 d9

d1 d2 together denote the city code,

d3 d4 d5 together constitute member number

d6 d7 d8 d9 together constitute group number

For example, if **141691334** is an ID number, **14** is the city code, **169** is the member number and **1334** is the group number.

The technical head plans on reassigning the employees to different divisions. To facilitate that, he decides to sort all the employees based on the group number followed by member number. However the input file "**file.txt**" obtained from the HR department contains multiple rows, segregated by the locality of the employee; i.e each row corresponds to a locality. Thus, he intends to do a two-pass sorting:

a) First sort each row of the file based on group number followed by member number using insertion sort
b) Then merge all the rows, again based on group number followed by member number so that all the employee ids in all the rows (localities) are sorted.

File **node.h** contains the definitions of the structures and global variables. These are required to read the file into lists and then to a global array. The file **file.txt** contains exactly **N** rows and every row contains an integer x (x < =M) followed x integers, all separated by comma. Both **N**, and **M** are preprocessor constants defined in **node.h** (by default set to 50 and 15 respectively).

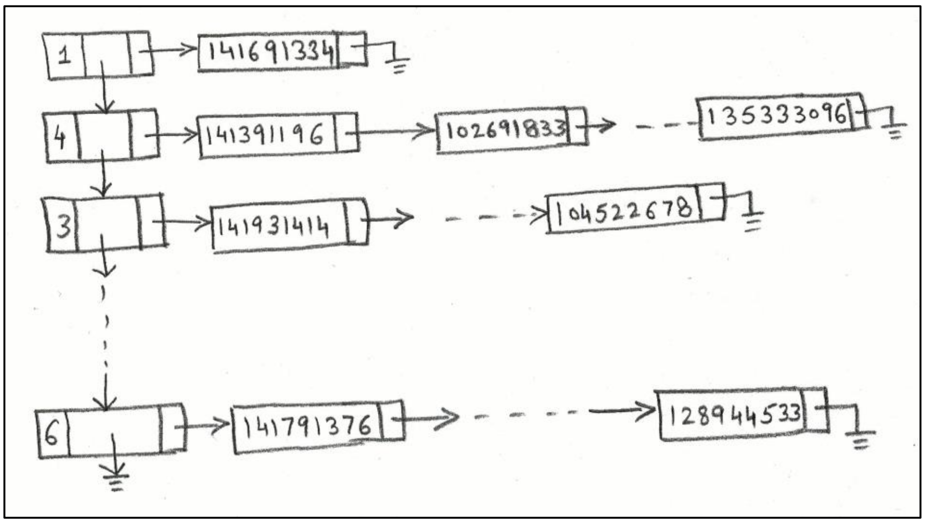
1. Implement the following function in **readdata.c** that takes the input filename as argument and returns a linked list of linked lists; i.e. it returns a linked list of Locality, wherein each element refers to one row (locality) in the input file. And each element of this list contains a linked list of members as well as the number of members in that locality. This function should read the file content exactly into the lists (linked list of linked lists) exactly in the order of its appearance in the file, with the complexity **O (M*N)**. [17M]

```
Locality * ReadFileintoLists(char *filename);
```

2. Implement the following function in **readdata.c** to print all the members in all localities (rows) where **start** points to the first locality

```
void PrintLists(Locality * start);
```
[Binary: 4M]

After reading the data (from file), the linked lists should look like as follows:



3. Implement the following function in **list2array.c** that converts the contents of the masterlist into **Arr** (global array). You should make use of the global array **Num_Elements[N]** to keep track of number of members in every row (locality).

```
void ConvertListstoArray(Locality * masterlist);
```
[Binary: 6M]

Note: If you are stuck in implementing functions `ReadFileintoLists()`, and/or `ConvertListintoArray()`, you may use the corresponding object files **readdata_obj.o** and **list2array_obj.o** at the cost of their credits to proceed further.

You may use the function `PrintArr()` that prints the global array **Arr** as utility function available in **main.c**.

4. Implement the following function in file **lower.c** that returns TRUE if `id1` is "lesser" than `id2`. Lesser is defined in terms of the group number followed by member number.

```
BOOL IsLower_GM( int id1, int id2);
```
[Binary: 5M]

5. Implement the following functions in file **insertion.c** which uses insertion sort to sort the id numbers based on group number followed by member number. [5M]

```
void InsertionSort_GM(int * row, int size);
```

The above function applies InsertionSort on a given row by sorting based on group number followed by member number

```
void InsertionSort_Arr();
```

The above function invokes the function `InsertionSort_GM()` to sort all rows of **Arr**.

6. Implement the following functions in **merge.c** to merge all sorted rows of **Arr**. [17M]

```
void Merge_GM(int * arr1, int * arr2, int * result);
```

The above function merges two arrays `arr1` and `arr2` based on group number followed by member number. This function is invoked by `Merge_Arr( )` to sort all rows of **Arr** and returns the finally sorted linear array of id numbers.

```
int * Merge_Arr();
```