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

Second Semester 2017-18   CS F111 Computer Programming

LABORATORY SESSION #13

(Linked List)

Consider the following type definition:

```
typedef struct student STUDENT;
typedef struct student
{
   char name[MAX];    // MAX is #defined to 40
   int count;
   int length;
   STUDENT *next;
}STUDENT;
```

A file named *roster.txt* present in the folder **/home/share** contains the first name of the all the students, each row containing a name; the first row however contains the number of students. Write a program to read the data from the file and create a linked list of unique names, by implementing the following functions:

a. Define a function **STUDENT * addName(STUDENT * list, char *name);** This function should create a new list if **list** is empty. Otherwise it should check for **name** in the linked list and if present, should increment the **count** accordingly. If **name** does not already exist in the list, the function should dynamically allocate memory and add a new element with name **name** at the end of the list, populate the individual members of the new structure element and return the new list.

b. Implement the function **void printUniqueNames(STUDENT *);** to print the list of unique names in the list along with its frequency. Use output redirection to redirect the output to the file *unique.txt*. How many unique namesyou're your function been able to find? Compare your output *unique.txt* with the contents in *names.txt* to verify your answer.

c. Implement the function **int findMostFrequent(STUDENT *);** which prints the most frequently occurring name in the list and returns its frequency.

d. To delete one student from the list, implement the following function:
   STUDENT * deleteSingle(STUDENT * list, char *name);
   This function should return the original list, if **name** is not present, decrement the **count** by one if **count** >1, or delete the node if **name** is present and **count** is equal to 1.

e. Modify the part (a) by implementing the following function:

   STUDENT * addSortOrder(STUDENT * list, char *name); This function while adding a name to the list will insert at the right spot so that the list is always in lexicographic ordering of the names.

f. Modify the structure definition by making the first name dynamically allocated and stored. In other words, the first field will be **char *nm;** and not an array.