CS29003 ALGORITHMS LABORATORY
Assignment No: 5
Last Date of Submission: 28–August–2014

In this assignment, you are supposed to implement a linear-time algorithm to sort student records with respect to the roll numbers. A student record consists of two fields: a roll number and a CGPA. An IITKGP undergraduate roll no is a string of length nine having the following format:

$d_0 d_1 D_2 D_3 d_4 d_5 d_6 d_7 d_8$

Here, d_i are decimal digits, whereas $D_2 D_3$ is a two-letter department code (in upper case). A CGPA is a floating-point number in the range [5,10].

In order to sort a list of records with respect to the *i*th digit, use a function that implements stable counting sort. Notice that a digit can take only ten values 0–9.

Also write a function to implement stable counting sort on the two-letter department code. IITKGP has only eighteen departments that can be stored in a sorted constant character array like the following:

```
#define NDEPT 18
const char DEPT[NDEPT][3] =
   { "AG", "AR", "BT", "CE", "CH", "CS", "CY", "EC", "EE",
     "GG", "HS", "IE", "MA", "ME", "MI", "MT", "NA", "PH" };
```

Instead of using two counting sorts based on upper-case letters, you should use a single counting sort based on the above sorted list. When you encounter a roll number in a student record, use binary search to locate the index of the department.

Write a *main*() function to do the following:

1. Read the number n of student records to be read.
2. Read n (RollNo,CGPA) pairs and store them in an array of student records.
3. Call digit-based counting sorts on the digits following the department code.
4. Call counting sort based on the department code.
5. Call digit-based counting sorts on the digits preceding the department code.
6. Print the array after every invocation of counting sort. In each line, print a roll number and the CGPA like:

   12CS30051, 8.57

Use only two arrays of size n. Stable counting sort requires the output to go to an array different from the input array. You do not need to copy back the output array after every invocation of counting sort. You should instead switch the roles (input and output) of your two arrays. For storing the counts in each counting sort, you need arrays of constant sizes (10 or 18).