

**Indian Institute of Technology, Guwahati**  
**Department of Computer Science and Engineering**  
**Data Structure Lab (CS210)**

**Assignment: 9**

**Date: 27<sup>th</sup> October, 2016.**

**Total Marks: 30 (Lab Assignment) + 30 (Offline assignment)**

**Deadline of Offline Assignment Submission: 2<sup>nd</sup> November, 2016.**

**Note:** Assignment 9 is optional. This is a makeup assignment for both Lab and Offline evaluations. It means marks of this lab will not be added to the total marks (of both lab and offline assignments). However, your score in assignment 9 will be added to your total obtained score (with limit maximum total marks). This is a chance to boost your score in Lab. Note that marks of lab assignments cannot be carried to the offline assignments and vice-versa.

This effectively means this lab is useless (in terms of your grade) for those who have already scored 100% till now and will continue to score 100% in all remaining assignments. However, I recommend all of you to attend and submit both parts of Assignment 9. This is a good practice session for all of you just before your final lab evaluation.

**Submission Instructions:**

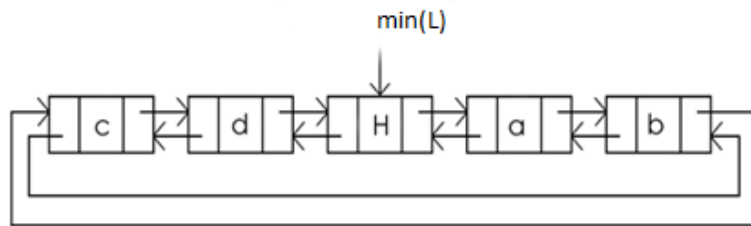
- Name of every file should be <Roll\_No>\_<Question\_No>.c. eg. 150101086\_1.c, 150101086\_2.c and so on.
- Name of the folder you upload should be <Roll\_No> eg. 154101086.
- Please do not upload files like a.out, desktop.ini, etc. If there are 'N' questions in assignment, there must be exactly 'N' ".c" or ".cpp" files in the folder. Use only zip to compress the folder to be uploaded.
- You will be awarded ZERO mark if the above rules are violated.
- Please note that you will not be able to submit once the dead line (with 12 hours' late submission) is over.

### Lab Assignment:

1. Define a structure which has following fields:

```
struct node
{
    int key;
    int degree;
    node* parent;
    node* child;
    node* left;
    node* right;
    bbool mark;
};
```

- a. **Insert\_Element(L, k):** Write a function that inserts a node with key k in a circular, doubly link list L. Assume that left and right pointers are used to create circular doubly link list. The other two pointers (that is, child and parent) are NIL here. For your purpose, you may assume degree is zero and mark is false for every node. Let min(L) be the pointer that points to the node with minimum key in L. The function returns the updated pointer min(L) after insertion.



- b. **Create\_List\_Random(L):** Write a function that creates a circular, doubly link list by repeatedly calling Insert\_Node() function starting from an empty list. The key will have some random values. Create two lists L1 and L2 using this function. Let min(L1) and min(L2) be two pointers that points to the node with minimum key in L1 and L2, respectively.
- c. **Concat\_Lists(L1, L2):** Write a function that concatenates two circular, doubly link lists L1 and L2 using their pointers min(L1) and min(L2), respectively.
- d. **Print\_List(L):** Write a function to print a circular doubly link lists. You need to print the [key, degree, mark] of each node.
- e. **Number\_of\_Nodes(L):** Write a function that returns the number of nodes in L.
- f. **Create\_Child(L1, L2):** Write a function that takes two lists (like L1 and L2 above) and connects L2 as a child of L1. The parent pointer of each node of L2 now points to min(L1). If the child pointer of L1 is NIL, then this child pointer will now point to min(L2). It may be noted that L1 may already have children. In this case, you need to concatenate L2 to the child list of L1. You need to increment the degree L1 by the size of L2.

g. **Heap\_Link(L, y, x):** Let y and x be pointing two nodes in the circular doubly link list L. You need to remove y from the list H. Make y a child of x by child and parent pointer of them accordingly. Increment degree of x by one. Make mark of y FALSE.

**Offline Assignment:**

2. Implement Fibonacci Heaps. You may use the above structure and functions to implement Fibonacci Heaps. You need to only Implement following operations:

- MAKE-FIB-HEAP
- FIB-HEAP-INSERT
- FIB-HEAP-EXTRACT-MIN
- FIB-HEAP-UNION

You can use the command interpreter that uses codes **c, i, e, u and S** to make, insert, extract the minimum, do a union operation and to show the heap structure, respectively.

Use test1.txt and test2.txt to test your code.