

# DATA STRUCTURES & ALGORITHMS 1

BATCH – B

[FRIDAY MARCH 01, 2019: 3:30 PM – 6:30 PM]

LAB ASSIGNMENT – 6

CODE:assign06

NOTES:

1. Please carefully read all assignments and there is **no choice**.
2. **Use the template for this assignment**
3. Each problem in this assignment has to be answered in the **same c file**.
4. Create a .c file following the **file name convention**:
  - a. If your roll number is 'abc' and assignment code is 'assignXX'. Then use the following file name convention as follows: 'abc-assignXX.c'
  - b. For example, if the roll number is 92 and assignment code is assign01, then the file name should be 092-assign01.c
  - c. Strictly follow the file name convention. When you are ready, submit the solution via google classroom.
5. **Follow variable and function naming conventions**
  - a. except for variables in for-loop, none of the other variables should be a single character.
  - b. The variable names and function names should indicate what they are storing/computing. For this assignment, we have given you some of the variable names and function names to use. They are highlighted as **function\_name** or **variable\_name**
  - c. All global variable should start with '**g\_**'
6. Indentation improves readability. Please pick an indentation style and **indent your code** appropriately.
7. Follow constants and type naming
  - a. All **constants** should be defined using **IFDEF and DEFINE**
  - b. All **structures** should have a **TYPDEF** to a simpler name
8. When in doubt about naming or style conventions, consult the following link:  
<https://users.ece.cmu.edu/~eno/coding/CCodingStandard.html>

---

Create a Structure '**student**' with the following details

1. Student name
2. Rollno (int is fine)
3. A pointer to 'next-student' (struct) instance

Most of the task today involves **creating and maintaining two linked list of students**. One for section-A and one for section-B. You have to maintain two head pointers corresponding to the two linked list. You might **be asked to merge the two** linked list to create a master linked list of students.

PROBLEMS [Total Marks: 20]:

You should implement the functions below, which is common for maintaining and modifying both Sec-A and Sec-B linked lists

1. [6 marks] Implement functions to create and print student instances
  - a. `create(*head)`: gets input from users (via `scanf`) and creates a student instance in Heap. It then **calls** `'add'` to add the new instance to the list pointed by `'head'`.
  - b. `print(*student)`: takes as input a student pointer and prints the detail
  - c. `print_from(*student)`: prints the linked-list starting from this node using recursion. It must use `print(*student)` function
  - d. `print_all(*head)`: prints the entire linked list pointed by `'head'`. It must use `print_from` function (Hint: pass the head pointer of the appropriate linked list to `print_from` and your job is done)
2. [6 marks] Implement the following functions(Warning: don't use `scanf` or `printf` inside any of the below functions.)
  - a. `add(*student, *head)`: takes two inputs: 1. Pointer to the new student instance 2. The head pointer for the appropriate linked list (sec-A or sec-B). The function adds the new student instance to the end of the list.
  - b. `update(i, *nstudent, *head)`: replaces the `'student'` at position `'i'` of the linked list with `'nstudent'`.
  - c. `insert(i, *student, *head)`: insert the student into the position `'i'` of the linked list.
  - d. `delete(i, *head)`: delete the student at index `i` of linked list.
  - e. `get(i, *head)`: return a pointer to the student at this index of the linked list pointed by `head`.(warning: make sure to check for a non-existing index `'i'` in all functions)
3. [Marks 8] Implement `'mergesort(*head)'`, which takes as input the head pointer of the linked list and sorts it by roll no.
  - a. Please make sure to implement `'merge'` as a separate function (we might use it outside of the merge-sort setting). It should take as input two sorted linked lists and merges them into one.
  - b. You should use recursion

The **switch case should have the following choices** which use the above functions. Please ask the user which linked list (sec A or sec B or master) they want to manipulate.

1. Create a new student (sec A or Sec B)
2. Insert a student (sec A or Sec B)
3. Delete a student (sec A or Sec B)
4. Create a master list (create a **master linked list** by merging sec A and sec B using merge)
5. mergesort a list (Sec A or Sec B or Master list)
6. Print all students (sec A or Sec B or Master list)

Evaluation Notes:

1. There are no tricky questions. This is a straightforward assignment.
  2. After asking you to merge Sec A and Sec B into master-list, we will not ask you to do any operations on Sec-A or Sec-B lists. So, don't worry about maintaining copies.
  3. We will not trick you with duplicate roll no. You can safely assume this and no need to write code for checking for duplicates.
-