DATA STRUCTURES & ALGORITHMS 1

BATCH – A [WEDNESDAY FEBRUARY 27, 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 IFNDEF and DEFINE
    b. All structures should have a TYPEDEF 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. CGPA
4. A pointer to 'next-student' (struct) instance

Create a global array called 'g_student_index'. The array stores pointers to Student structure. Define MAX_STUDENTS = 50 and use that as the size of the array.

PROBLEMS [Total Marks: 20]:

Most of the task today involves **creating and maintaining a linked list of students**. In addition, you will be **maintaining a sorted index to implement binary search** by rollno. Please assume that entering unique roll_no is the responsibility of the user. So, don't worry about verifying the uniqueness of rollno.

1. [8 marks] Implement functions to create and print student instances
    a. create(): gets input from users (via scanf) and creates a student instance in Heap. It returns the pointer for the newly created student.
    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(): prints the entire linked list. It must use print_from function (Hint: pass the head pointer to print_from and your job is done)

2. [8 marks] Implement the following List ADT functions(Warning: don't use scanf or printf inside any of the below functions.)
    a. add(*student): take a student pointer as input and add it to the end of the list.
    b. update(i, *nstudent): replace the 'student' at position 'i' with 'nstudent'.
    c. insert(i, *student): insert the student into the position 'i'
    d. delete(i): delete the student at index i
    e. get(i): return a pointer to the student at this index

    (warning: make sure to check for a non-existing index in all functions)

3. [Marks 4] Implement the following functions to maintain g_student_index
    a. Maintaining index (1 mark)
        i. add_to_index (*student): you should call this to add a newly created student to the array. You must call this from create.
        ii. delete_from_index (*student): call this when a student is deleted
        iii. Use both add_to_index and delete_from_index for the case of updating a student.
    b. update_index: sorts the array based on roll_no. You can use any sorting algorithms, don't worry about their time complexity. (1 mark)
    c. search(roll no): implement binary search to search for a student based on roll no. (1 mark). If the student exists, print the details using a call to the 'print' function you wrote earlier.
    d. Print_by_index: instead of following the linked list chain. It just prints all students according to the g_student_index array. This must use the 'print' function you wrote earlier (1 mark)

The switch case should have the following choices which use the above functions

1. Create a new student
2. Print all students
3. Update a student

4. Insert a student
5. Delete a student
6. Update index array
7. Search by roll no
8. Print by index array
