CSE108
LW 12

- **Using mobile phones, flash disks, internet and any other record or communication media is strictly forbidden during lab sessions. Throughout a lab session, all such media must be kept turned off and in a closed environment. Violation of this rule is punished with a grade 0, -100 or worse. Before doing anything else, make sure that your computer is not attached any such media.**
  Make sure that you have deleted all of your work PARMANENTLY before leaving the first sessions.

PART-I (2 PT)

Define a structure, **student_t**, to hold a student id (integer) and her grade (integer).

- Write a function, **create_records**, which:
  - Takes an integer (input argument) to represent number of students, n,
  - Takes a file name (input argument) and create a binary file with this name
  - Without using arrays, creates and saves n student_t objects into the binary file such that:
    - id fiels are assigned starting from 1 and increased by 1: 1, 2, 3, …., n
    - grades are assigned randomly between 40 and 90 is to each student

  void create_records(int n, const char* file_name)

node_t
id: int name: char[20] grade: int next: node_t*

- Write a function, **read_records**, which takes a file name and reads all student_t records in it into a linked-list using the node_t structure. Do not forget to assign a NULL pointer to the name field so that print function can understand a name has not been assigned yet and do not try to print name field, so segmentation fault can be avoided.
- Write a function, **print_records**, which takes a linked-list built by read_records() and prints all valid fields of students.

PART-II (3 PT)

Implement the following functions:

- **void add_name(node_t* list, int id, const char* name)**: adds the name to the student having the given id.
- **int free_list(node_t* list)**: (Recursive) Frees all dynamically allocated memory and returns the number of deleted nodes.

PART-III (2 PT)

Implement the following function:

- **int delete_fails(node_t** list, int pass_grade)** : takes students and passing grade, deletes the failed students from the list and returns the number of failed students.

BONUS (+1): Write delete_fails() using recursion.

PART-IV (2 PT)

Implement the following function:

- **int insert_list(node_t** list1, node_t* list2, int id)** :
    - if a student having given id exists in list1, inserts list2 into list1 just after the student and returns 1,
    - if id is -1, inserts all elements of list2 before all elements of list1 and returns 1,
    - else, does nothing and returns 0.

Build a new linked-list using the functions in part1 and insert the new list into the old one using the function above.