CS19003 Programming and Data Structures Laboratory
Assignment 7

Name the files as ⟨assignment no⟩_⟨question no⟩_⟨roll no⟩.c, without the ⟨ and ⟩. Consult your mentor for any confusion. Penalty if the file names do not stick to this convention.

Please take note of the following:

- ▷ The structure definitions/declarations and names of variables and functions should be exactly as given in the questions. Any deviation will attract penalty.
- ▷ The format of the output should match as closely with the sample output as possible. Significant deviation will attract penalty. In the event of any confusion, ask us.
- ▷ Global variables (including structure variables) are NOT allowed. However, the structure definitions will be global.
- ▷ String library functions covered in the PDS lecture slides are allowed. You may use malloc.h or stdlib.h for using the *malloc* and *free* functions. No other library is allowed.
- ▷ Read the questions carefully. Any deviation from the specifications mentioned will attract penalty. In the event of any confusion, ask us.

---

1. You will rewrite the interactive program to view, insert, search for and delete student records that you wrote in the last lab test, using **linked list**.

   - ▷ Define the following global structure. The details of the fields are written as comments in the definiton.

     struct student{
     *char name[20]; /** only English letters, no space */
     *int id; /** An integer between 0 and 1000, inclusive. These values are unique. */
     *struct student *next;* };

     [Note: The optional member of the structure in LT2 is not there any more.]
   - ▷ Make the following declaration in *main*. The variable *stlist* will be the head pointer of the linked list. Initialize an empty linked list with *stlist* as head.

     *struct student *stlist;*
   - ▷ Write a function named *View* with appropriate prototype that displays the details of all the student records in a linked list. The records may be displayed in any order (not necessarily in the same the order as in the sample outputs). Make sure you handle the case that the database is empty (see sample input/output).

- ▷ Write a function named *Insert* with appropriate prototype that asks the user for details of a new student, inserts a new student record in a linked list if a student with the same id is not present and prints a suitable message otherwise (see sample input/output).
- ▷ Write a function named *Search* with appropriate prototype that asks the user for a student's id as input, searches a linked list, and either reports that there is no entry with that id, or displays the name of the student with that id.
- ▷ Write a function named *Delete* with appropriate prototype that asks the user for a student's id as input, searches a linked list, and either reports that there is no entry with that id, or deletes the student record with that id from the linked list. Remember to *free* the deleted node.
- ▷ In *main*, display the following menu.

  Enter your option:

  1: Insert record:

  2. View record:

  3. Delete record:

  4. Search record:

  5. Exit

  If the user presses 1,2, 3 or 4, accomplish the requested task by calling the appropriate function and then return to the above menu. If the user presses 5, the program terminates. If the user presses anything else, display "Invalid input" and return to the menu.

  Note: You may define functions in addition to the ones specified above.

Sample input/output:

Sample input/output:

Enter your option:

1: Insert record

2. View record

3. Delete record

4. Search record

5. Exit

7

Invalid input.

Enter your option:

1: Insert record

2. View record

3. Delete record

4. Search record

5. Exit

2

Database empty.

Enter your option:

1: Insert record
2. View record
3. Delete record
4. Search record
5. Exit
1
Enter student's ID: 34
Enter student's name: Tushar
Enter your option:
1: Insert record
2. View record
3. Delete record
4. Search record
5. Exit
2
Name: Tushar, ID: 34
Enter your option:
1: Insert record
2. View record
3. Delete record
4. Search record
5. Exit
1
Enter student's ID: 34
Record already present.
Enter your option:
1: Insert record
2. View record
3. Delete record
4. Search record
5. Exit
1
Enter student's ID: 76
Enter student's name: Rustam
Enter your option:
1: Insert record

2. View record
3. Delete record
4. Search record
5. Exit
1
Enter student's ID: 56
Enter student's name: Tanisha
Enter your option:
1: Insert record
2. View record
3. Delete record
4. Search record
5. Exit
2
Name: Tanisha, ID: 56
Name: Rustam, ID: 76
Name: Tushar, ID: 34
Enter your option:
1: Insert record
2. View record
3. Delete record
4. Search record
5. Exit
3
Enter student's ID: 45
Record absent.
Enter your option:
1: Insert record
2. View record
3. Delete record
4. Search record
5. Exit
3
Enter student's ID: 34
Enter your option:
1: Insert record

2. View record
3. Delete record
4. Search record
5. Exit
2
Name: Tanisha, ID: 56
Name: Rustam, ID: 76
Enter your option:
1: Insert record
2. View record
3. Delete record
4. Search record
5. Exit
4
Enter student's ID: 79
Record absent.
Enter your option:
1: Insert record
2. View record
3. Delete record
4. Search record
5. Exit
4
Enter student's ID: 56
Record present. Student's name: Tanisha
Enter your option:
1: Insert record
2. View record
3. Delete record
4. Search record
5. Exit
3
Enter student's ID: 56
Enter your option:
1: Insert record
2. View record

3. Delete record

4. Search record

5. Exit

3

Enter student's ID: 56

Record absent.

Enter your option:

1: Insert record

2. View record

3. Delete record

4. Search record

5. Exit

2

Name: Rustam, ID: 76

Enter your option:

1: Insert record

2. View record

3. Delete record

4. Search record

5. Exit

5

[100 points]

2. **Insertion sort in linked list of words.**

   ▷ Make the following global structure definition:
   struct word{
   *char s[20]; /** only English letters, no space *\/*
   *struct word *next;* };

   ▷ In main, make the following declaration:
   *struct word *H;*
   This is the head pointer of the linked list of words that you will build. Initialize an empty linked list with *H* as head.

   ▷ Write a function *insert* with appropriate prototype that does the following. Assume that the linked list of words with head *H* is sorted alphabetically (note: an empty list is sorted). The function insert asks the user for an English word. Assume that the word is a sequence of at most 19 English lower-case letters (no space). Then it inserts the word at an appropriate position in the linked list such that the resultant list is alphabetically sorted.

▷ In main, display the following:

Enter 1 to type a new word, 0 to quit:

Assume that the user always enters either 1 or 0 (so you don't have to check for invalid input). If the user presses 1, call the insert function and add a new word to the list, and then display the above line back. If the user presses 0, print the list from head to tail and terminate. Print each new word in a new line. If your program is correct, the list of words printed will be alphabetically sorted. Note that there may be multiple occurences of a single word in the list (see sample output).

[Note: the function insert makes changes to the list, and those changes should be visible in the main. So, decide the argument type(s) of the function accordingly.]

Sample input/output:

Enter 1 to type a new word, 0 to quit: 1

Enter word: tina

Enter 1 to type a new word, 0 to quit: 1

Enter word: wizard

Enter 1 to type a new word, 0 to quit: 1

Enter word: apple

Enter 1 to type a new word, 0 to quit: 1

Enter word: tina

Enter 1 to type a new word, 0 to quit: 1

Enter word: tiny

Enter 1 to type a new word, 0 to quit: 1

Enter word: sun

Enter 1 to type a new word, 0 to quit: 1

Enter word: sunny

Enter 1 to type a new word, 0 to quit: 0

apple

sun

sunny

tina

tina

tiny

wizard

[100 points]