Department of Information Systems and Technologies

CTIS 152 – Data Structures and Algorithms
FALL 2024 - 2025

Lab Guide #20 – Week 13 – 2

OBJECTIVE : Linked List Operations and Exercises

Instructor : Burcu LIMAN
Assistant : Berk ÖNDER, Engin Zafer KIRAÇBEDEL

Q1. You will implement the **linkedList_struct.h** file to store the information of the customers in an insurance company. The list will store the **customer id, customer surname** and **entrance year** for several insurance policies.

Write the following functions;

- **`createList`:** takes the file pointer as input parameter, creates and initializes the linked list with the values given in the file (customer id, customer surname and entrance year for several student).
- **`deletePolicy`:** gets the initial address of a list as parameter and removes the policies before year 2020. The function returns the initial address of the list.

Write a C program that will test your functions and remove the policies before year 2020.

Project Name: LG20_Q1
File Name: Q1.cpp

Example Run:

```
Original List:
813 Miller 2019 -> 516 Vargas 2020 -> 114 Evans 2021 -> 123 Diaz 2018 -> 151
Kochhar 2022 -> 631 DeHaan 2021 -> 745 Richins 2017 -> NULL

*** Policies before year 2020 will be deleted ***

516 Vargas 2020 -> 114 Evans 2021 -> 151 Kochhar 2022 -> 631 DeHaan 2021 ->
NULL
```

policies.txt

```
813 Miller 2019
516 Vargas 2020
114 Evans 2021
123 Diaz 2018
151 Kochhar 2022
631 DeHaan 2021
745 Richins 2017
```

Q2. An email service keeps the list of login information (username, city and last login) which may have suspicious activities in the text file **"alert.txt"**, warns the users showing their login history, if there is no suspicious activities then removes the login information for the specified user from the list. Otherwise displays a warning message.

The program reads the information from the text file into a linked list. Displays the alert login history for the specified user, removes all login history from the linked list or displays a warning message depending on the user's answer.
You are supposed to create a structure linked list header file (named **linkedList_struct.h**) with the following functions; **`getNode`, `addAfter`, `addBeginning`, `displayList`, `deleteAfter`, `deleteFirst`.**

Implement the following functions in your source program and make the necessary validity checks;

- **`createList`:** gets the file pointer as input parameter, reads the login information for several user from the text file and creates the linked list. The function returns the initial address of the linked list. (You may read the date and the time into a string). Sample line from the file: `carolina,basildon,01.02.2022 06:08`
- **`searchForAUser`:** gets the initial address of the list and the username to be searched as input parameters, displays all the login information for the specified user. The program returns 1 if the given username found in the list, otherwise returns 0.
- **`deleteUserLogins`:** gets the initial address of the list and the username to be deleted as input parameters, deletes all the login information for the specified user from the list. The function returns the initial address of the linked list.

Project Name: LG20_Q2
File Name: Q2.cp

alert.txt

```
carolina,basildon,01.02.2022 06:08
gouya,echuca,01.03.2022 14:22
carolina,cahaba heights,01.04.2022 12:45
adam,martin,01.04.2022 20:01
renee,tel aviv,01.04.2022 22:10
aidan,chatou,01.04.2022 01:17
carolina,new york,02.04.2022 04:59
sean,shavano park,01.04.2022 09:05
adam,martin,01.04.2022 21:09
carolina,eagle,01.04.2022 10:05
```

Example Run #1:

```
Login information for the users who may have suspicious activites:

User Name       City            Last Login
***********     ********        ****************
carolina        basildon        01.02.2022 06:08 ->
gouya           echuca          01.03.2022 14:22 ->
carolina        cahaba heights  01.04.2022 12:45 ->
adam            martin          01.04.2022 20:01 ->
renee           tel aviv        01.04.2022 22:10 ->
aidan           chatou          01.04.2022 01:17 ->
carolina        new york        02.04.2022 04:59 ->
sean            shavano park    01.04.2022 09:05 ->
adam            martin          01.04.2022 21:09 ->
carolina        eagle           01.04.2022 10:05 ->
NULL

Please enter your account name to see the login history: tony
There is no account with the name tony!
```

Example Run #2:

```
Login information for the users who may have suspicious activites:

User Name       City            Last Login
***********     ********        ****************
carolina        basildon        01.02.2022 06:08 ->
gouya           echuca          01.03.2022 14:22 ->
carolina        cahaba heights  01.04.2022 12:45 ->
adam            martin          01.04.2022 20:01 ->
renee           tel aviv        01.04.2022 22:10 ->
aidan           chatou          01.04.2022 01:17 ->
carolina        new york        02.04.2022 04:59 ->
sean            shavano park    01.04.2022 09:05 ->
adam            martin          01.04.2022 20:01 ->
carolina        eagle           01.04.2022 10:05 ->
NULL

Please enter your account name to see the login history: carolina

carolina        basildon        01.02.2022 06:08 ->
carolina        cahaba heights  01.04.2022 12:45 ->
carolina        new york        02.04.2022 04:59 ->
carolina        eagle           01.04.2022 10:05 ->

Is there any suspicious account activity? (y/n): n
** All the login info will be deleted ***

Login history updated!

User Name       City            Last Login
***********     ********        ****************
gouya           echuca          01.03.2022 14:22 ->
adam            martin          01.04.2022 20:01 ->
renee           tel aviv        01.04.2022 22:10 ->
aidan           chatou          01.04.2022 01:17 ->
sean            shavano park    01.04.2022 09:05 ->
adam            martin          01.04.2022 21:09 ->
NULL
```

Q3.

Create a nested structure **applicantsOfII** and **grades** as follows:

```
typedef struct{
        int englishProficiency, jury, graduateExam;
 } grades_t;

 typedef struct{
      int id;
      grades_t gr;
      double overall;
}applicantsOfII_t;
```

Write the following functions:

- **readFile,** which gets a set of application information from a text file named **applicants.txt** until the end of the file is reached, also returns the size of the structure array (Do not forget to initialize the overall grade to 0 for each student).
- **calculate,** which calculates the overall applicants' grades' average and the overall grade of each applicant with the loads of English proficiency being 30%, jury being 50%, and the graduate exam being 20%)
- **display,** which displays the content of the structure array of **applicantsOfII_t** type.
- **findPassFail,** a function that finds and displays the number of the applicants who fail and pass the elimination as well as displaying the average of all applicants' grades'. (An applicant passes if overall >= average, otherwise student fails).

Write a C program that reads the entirety of applicants' information from applicants.txt file into an array of structures, and displays all the information on the screen as necessary, as shown in the example run below.

Project Name: LG20_Q3
File Name: Q3.cpp

applicants.txt

```
1222 45 67 98
1333 89 45 33
1444 67 76 99
```

Example Run:

```
Applicant ID: 1222
Scores:
Applicant English Proficiency: 45
Applicant Jury: 67
Applicant Graduate Examination: 98
Applicant Overall: 66.6

Applicant ID: 1333
Scores:
Applicant English Proficiency: 89
Applicant Jury: 45
Applicant Graduate Examination: 33
Applicant Overall: 55.8

Applicant ID: 1444
Scores:
Applicant English Proficiency: 67
Applicant Jury: 76
Applicant Graduate Examination: 99
Applicant Overall: 77.9

Average is 66.8
Number of the applicants who pass is 1
Number of the applicants who fail is 2
```

Additional Questions

AQ1.

Write a C program that reads the student information (consisting of id and cgpa) from the file named **"students.bin"** and stores them in a linked list. Then the program reverses the linked list recursively without using a stack.

Example Run:

```
Content of the file
20036521 2.35 -> 20115632 3.28 -> 20213658 3.04 -> 20496253 2.02 -> 20541236 2.56 ->
20365412 2.25 -> 20695147 3.65 -> NULL

Reverse of the list
20695147 3.65 -> 20365412 2.25 -> 20541236 2.56 -> 20496253 2.02 -> 20213658 3.04 ->
20115632 3.28 -> 20036521 2.35 -> NULL
```

AQ2.

Define a structure named **vegAndFruit_t** with the fields; *id*, *name*, and *price*.
Write the following functions;

- **displayList:** displays the vegetable information in a linked list on the screen as shown in the example run.
- **menu:** displays a menu on the screen with the options seen in the example run.
- **searchNode:** gets the initial address of a vegetable list and an integer id as parameters, and searches the id through the list. If the id is found in the list, returns the address of the node, otherwise returns NULL.
- **addToEnd:** gets the initial address of a list and a vegetable information as parameters, and adds the new vegetable information to the end of the list. Used for "Add Record" functionality.
- **createList:** gets an input text file pointer as parameter, reads several vegetable information from the text file, stores them in a linked list, and returns the initial address of the linked list.
- **updateList:** gets the initial address of a list, the address of a node whose data will be updated and the field number (1.Id, 2.Name, 3.Price) as parameters, and updates the specified field with the information taken from the user. If the id field is going to be updated, then the function should check whether the new id already exists in the list or not. If it does not exist, the function updates the id field. Otherwise, gives a warning message specifying the id already exists in the list.
- **writeList:** writes the vegetable info in a linked list to a binary file for which a name is given as a parameter.

Write a C program that reads several vegetable information from a text file whose name is given by the user, creates a linked list, and displays the list. Then the program will display a menu on the screen with the options **Add Record**, **Update Record**, and **Exit,** and perform the chosen operation by using the necessary functions. The program should write the last form of the list back to a binary file named the same as the file read but with a .bin extension instead of .txt, with a single fwrite function. Examine the example run very carefully.

Example Run:

```
Enter the text file name: input.txt
Enter the text file name: vegandfruit.txt

ID     Name         Price
-------------------------
118    Tomato       3.00
111    Potato       2.50
123    Spinach      2.25
712    Aubergine    3.50
455    Cucumber     1.50
955    Apple        4.25
166    Bean         5.00

   MENU
----------------
1. Add Record
2. Update Record
3. Exit
Enter your choice: 1

Enter vegetable Id   : 166
The ID: 166 already exists.

Enter vegetable Id   : 198
Enter vegetable name : Onion
Enter vegetable price: 2.75

ID     Name         Price
-------------------------
118    Tomato       3.00
111    Potato       2.50
123    Spinach      2.25
712    Aubergine    3.50
455    Cucumber     1.50
955    Apple        4.25
166    Bean         5.00
198    Onion        2.75

   MENU
----------------
1. Add Record
2. Update Record
3. Exit
Enter your choice: 2

Enter vegetable Id   : 198
198    Onion        2.75

Update (1. ID, 2. Name, and 3. Price): 1
Enter the new Id: 152

ID     Name         Price
-------------------------
118    Tomato       3.00
111    Potato       2.50
123    Spinach      2.25
712    Aubergine    3.50
455    Cucumber     1.50
955    Apple        4.25
166    Bean         5.00
152    Onion        2.75

     MENU
----------------
1. Add Record
2. Update Record
3. Exit
Enter your choice: 2

Enter vegetable Id   : 152
152    Onion        2.75

Update (1. ID, 2. Name, and 3. Price): 2
Enter the new name: Orange
```

```
ID     Name         Price
-------------------------
118    Tomato       3.00
111    Potato       2.50
123    Spinach      2.25
712    Aubergine    3.50
455    Cucumber     1.50
955    Apple        4.25
166    Bean         5.00
152    Orange       2.75

   MENU
----------------
1. Add Record
2. Update Record
3. Exit
Enter your choice: 2

Enter vegetable Id   : 152
152    Orange       2.75

Update (1. ID, 2. Name, and 3. Price): 3
Enter the new price: 4

ID     Name         Price
-------------------------
118    Tomato       3.00
111    Potato       2.50
123    Spinach      2.25
712    Aubergine    3.50
455    Cucumber     1.50
955    Apple        4.25
166    Bean         5.00
152    Orange       4.00

   MENU
----------------
1. Add Record
2. Update Record
3. Exit
Enter your choice: 3
Writing linkedlist to binary file "vegandfruit.bin"
Goodbye
```