Assignment 2 Self-referential structures, structures, pointer arrays

Due: Nov 30 (Tue) 11:59 pm Total marks: 140 pts

Note that unlike the weekly labs, this is an individual work, and you should not discuss with others. Seeking help from current and previous students will result in a 0. Ask the instructor for help.

In this assignment you are going to implement a customized data structure and then implement a self-referential data structure.

Question 1 pointer arrays, structures in C (80 pts)

Subject: Structures, array of pointer to structures

Specification: in this exercise you are going to develop a simple patient database management system for a hospital.

The purpose of this exercise is to help you get better understanding of some fundamental concepts in C that we covered after midterm. These include array of pointers, structures, pointer to structures, dynamic memory allocation, formatted IO.

Download file `a2DB.c` to start off. Study the provided codes.

The database maintains a collection of patient records. Each patient record is naturally implemented as a structure, and contains the following information fields:

- `name` of type `char []`, which represents the patient's name
- `int age`, which represents the patient's age
- `float highBP`, which represents the patient's high (systolic) blood pressure
- `float lowBP`, which represents the patient's low (diastolic) blood pressure
- `float riskFactor`, which represents the patient's risk factor for stroke due to hypertension.

Programmatically, the database maintains an **array of pointers to record structs,** as shown in the Figure below.

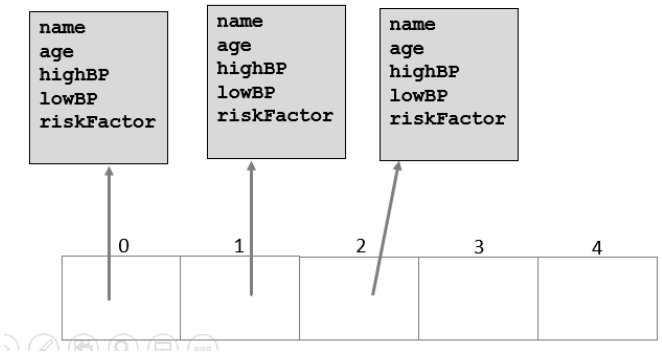


Figure 1 Main database: Array of pointers to record (structure)

The program will provide several basic functions:

- Accepting entry of a new patient record into the current database, and generating risk factor for the patient.
- Displaying all patient records in the current database
- Removing an existing patient's record in the current database
- Swapping the records in the current database
- Sorting the patient records in the current database
- Clearing the current database

Specifically, the program keeps on prompting the user with the following menu, until `q` or `Q` is chosen, which terminates the program.

The program should fulfill the following functionalities (some have been implemented for you):

- Keeps on prompting and responding to user inputs. Valid input includes `V/v`, `N/n`, `D/d`, `S/s`, `C/c`, `R/r`, `P/p` and `Q/q`. Displays error messages for other inputs as shown below. (This has been implemented for you.)

```
red 325 % gcc a2DB.c
red 326 % a.out
-----------------------------------------------------------------
|    (N)ew record        (R)emove record     Swa(p) records     |
|    (S)ort database     (C)lear database    (D)isplay db       |
|    (Q)uit              *Case Insenstive*                      |
-----------------------------------------------------------------
choose one: x
not a valid input!

-----------------------------------------------------------------
|    (N)ew record        (R)emove record     Swa(p) records     |
|    (S)ort database     (C)lear database    (D)isplay db       |
|    (Q)uit              *Case Insenstive*                      |
-----------------------------------------------------------------
choose one: sort
not a valid input!

-----------------------------------------------------------------
|    (N)ew record        (R)emove record     Swa(p) records     |
|    (S)ort database     (C)lear database    (D)isplay db       |
|    (Q)uit              *Case Insenstive*                      |
-----------------------------------------------------------------
choose one:
```

- When the users enters `n` or `N`, which represents `New record`, the program further prompts the user to enter a new patient record into the database. The function reads in patient's name, age, high blood pressure and low blood pressure. It then calculates and sets the risk factor of the new patient. The function then creates a new patient record and inserts the new record into the database (i.e., the pointer array).

  Assume in the input, the high blood pressure is higher than the low blood pressure. Assume the optimal high blood pressure (optimalH) is 120 and optimal low blood pressure (optimalL) is 80. The risk factor is evaluated as follows (the lower the healthier):

- The risk factor is 0 if both the blood pressures are below (or equal to) their optimal value. That is, the high blood pressure is not higher than optimalH and low blood pressure is not higher than optimalL.
  - Here we assume that from the perspective of stroke risk, it is always good that blood pressure is low. Thus a high and low blood pressure of 45 and 30 respectively also get risk factor 0 (which means low risk for stroke, but in reality such person might have other problem.)
- If one of blood pressure is below (or equal to) to its optimal value, but the other is above the optimal value by no more than 10, then the risk factor is 1;
- If both the pressures are above their optimal value (but) by no more than 10, then the risk factor is 2;
- If one pressure is below (or equal to) its optimal value, but the other is above the optimal value by 11-20, then the risk factor is 3;
- If both the pressures are higher than their optimal value by 11-20, the risk factor is 4;
- In all other higher cases, the risk factor is 5.

The above evaluation criterion applies to patients who are no more than 50 years of age. If the patient is more than 50 years of age, then each case above has additional risk factor of 0.5, except the first case where both the blood pressure are below (or equal to) their optimal values.

- When the user enters `d` or `D`, displays the current database of (all) patient records.
- When the user enters `c` or `C`, clear the current database (partially implemented).

Sample inputs/outputs involving `n/N` and `d/D` and `c/C` are shown below.

```
-----------------------------------------------------------------
|    (N)ew record        (R)emove record     Swa(p) records     |
|    (S)ort database     (C)lear database    (D)isplay db       |
|    (Q)uit              *Case Insensitive*                     |
-----------------------------------------------------------------
choose one: d
==============================
========== 0 records ==========

-----------------------------------------------------------------
|    (N)ew record        (R)emove record     Swa(p) records     |
|    (S)ort database     (C)lear database    (D)isplay db       |
|    (Q)uit              *Case Insensitive*                     |
-----------------------------------------------------------------
choose one: n
name: Judy Sue
age: 30
high bp: 110
low bp: 88

-----------------------------------------------------------------
|    (N)ew record        (R)emove record     Swa(p) records     |
|    (S)ort database     (C)lear database    (D)isplay db       |
|    (Q)uit              *Case Insensitive*                     |
```

```
-----------------------------------------------------------------
choose one: d
================================

name:   Judy Sue
age:    30
bp:     110.00  88.00
risk:   1.0

========== 1 records ==========
-----------------------------------------------------------------
|    (N)ew record        (R)emove record     Swa(p) records     |
|    (S)ort database     (C)lear database    (D)isplay db       |
|    (Q)uit              *Case Insensitive*                     |
-----------------------------------------------------------------
choose one: n
name: Cindy White
age: 52
high bp: 100.7
low bp: 89.4

-----------------------------------------------------------------
|    (N)ew record        (R)emove record     Swa(p) records     |
|    (S)ort database     (C)lear database    (D)isplay db       |
|    (Q)uit              *Case Insensitive*                     |
-----------------------------------------------------------------
choose one: d
================================

name:   Judy Sue
age:    30
bp:     110.00  88.00
risk:   1.0

name:   Cindy White
age:    52
bp:     100.70  89.40
risk:   1.5

========== 2 records ==========

-----------------------------------------------------------------
|    (N)ew record        (R)emove record     Swa(p) records     |
|    (S)ort database     (C)lear database    (D)isplay db       |
|    (Q)uit              *Case Insensitive*                     |
-----------------------------------------------------------------
choose one: c
are you sure to clear db? (y) or (n)? y

-----------------------------------------------------------------
|    (N)ew record        (R)emove record     Swa(p) records     |
|    (S)ort database     (C)lear database    (D)isplay db       |
|    (Q)uit              *Case Insensitive*                     |
-----------------------------------------------------------------
choose one: d
================================
========== 0 records ==========
```

```
-----------------------------------------------------------------
|    (N)ew record        (R)emove record     Swa(p) records     |
|    (S)ort database     (C)lear database    (D)isplay db       |
|    (Q)uit              *Case Insensitive*                     |
-----------------------------------------------------------------
choose one:
```

Implement function `void enterNew (struct db_type * pArr[])`, `void displayDB(struct db_type * pArr[])` and `clearDB(struct db_type * pArr[])` to accomplish the above. Feel free to add any helper functions as needed.

- When user chooses `R` or `r`, which represents `Remove record`, further prompts the user for the name of the patient whose record is to be removed (case sensitive). If no record by that name is found in the current database, then the error message "record not found" should be printed out. If the record is found, the record is removed from the database, with message "record [xx] removed successfully". **Note that after removal, the relative ordering of the remaining records should remain unchanged, as shown below.**

```
-----------------------------------------------------------------
|    (N)ew record        (R)emove record     Swa(p) records     |
|    (S)ort database     (C)lear database    (D)isplay db       |
|    (Q)uit              *Case Insensitive*                     |
-----------------------------------------------------------------
choose one: d
==============================

name:   Alice Zue
age:    20
bp:     110.00  76.00
risk:   0.0

name:   Bill Las
age:    21
bp:     130.00  83.00
risk:   2.0

name:   Cindy Sue
age:    33
bp:     134.00  90.70
risk:   4.0

name:   Dusan Luc
age:    60
bp:     168.50  130.20
risk:   5.5

========== 4 records ==========

-----------------------------------------------------------------
|    (N)ew record        (R)emove record     Swa(p) records     |
|    (S)ort database     (C)lear database    (D)isplay db       |
|    (Q)uit              *Case Insensitive*                     |
-----------------------------------------------------------------
choose one: r
enter full name to remove: Linh Ng
record not found
```

```
-------------------------------------------------------------------
|    (N)ew record        (R)emove record     Swa(p) records     |
|    (S)ort database     (C)lear database    (D)isplay db       |
|    (Q)uit              *Case Insensitive*                     |
-------------------------------------------------------------------
choose one: r
enter full name to remove: Bill las
record not found

-------------------------------------------------------------------
|    (N)ew record        (R)emove record     Swa(p) records     |
|    (S)ort database     (C)lear database    (D)isplay db       |
|    (Q)uit              *Case Insensitive*                     |
-------------------------------------------------------------------
choose one: r
enter full name to remove: Bill Las
record [Bill Las] removed successfully.

-------------------------------------------------------------------
|    (N)ew record        (R)emove record     Swa(p) records     |
|    (S)ort database     (C)lear database    (D)isplay db       |
|    (Q)uit              *Case Insensitive*                     |
-------------------------------------------------------------------
choose one: d
==============================

name:   Alice Zue
age:    20
bp:     110.00  76.00
risk:   0.0

name:   Cindy Sue
age:    33
bp:     134.00  90.70
risk:   4.0

name:   Dusan Luc
age:    60
bp:     168.50  130.20
risk:   5.5

========== 3 records ==========

-------------------------------------------------------------------
|    (N)ew record        (R)emove record     Swa(p) records     |
|    (S)ort database     (C)lear database    (D)isplay db       |
|    (Q)uit              *Case Insensitive*                     |
-------------------------------------------------------------------
choose one: r
enter full name to remove: Cindy
record not found

-------------------------------------------------------------------
|    (N)ew record        (R)emove record     Swa(p) records     |
|    (S)ort database     (C)lear database    (D)isplay db       |
|    (Q)uit              *Case Insensitive*                     |
-------------------------------------------------------------------
choose one: R
```

```
enter full name to remove: Cindy Sue
record [Cindy Sue] removed successfully.

-----------------------------------------------------------------
|    (N)ew record        (R)emove record     Swa(p) records     |
|    (S)ort database     (C)lear database    (D)isplay db       |
|    (Q)uit              *Case Insensitive*                     |
-----------------------------------------------------------------
choose one: d
================================

name:   Alice Zue
age:    20
bp:     110.00  76.00
risk:   0.0

name:   Dusan Luc
age:    60
bp:     168.50  130.20
risk:   5.5

========== 2 records ==========

-----------------------------------------------------------------
|    (N)ew record        (R)emove record     Swa(p) records     |
|    (S)ort database     (C)lear database    (D)isplay db       |
|    (Q)uit              *Case Insensitive*                     |
-----------------------------------------------------------------
choose one:
```

Implement function `void remove(struct db_type * pArr[])` to accomplish the above.

- When the user chooses `P` or `p`, which represents `Swap records`, then starting from the 1st record, swaps the pairs of adjacent records. That is, the 1st record swaps with the 2nd , the 3rd swaps with the 4th ……. If the total number of records is an odd number, then the last record is not swapped with any records.

```
-----------------------------------------------------------------
|    (N)ew record        (R)emove record     Swa(p) records     |
|    (S)ort database     (C)lear database    (D)isplay db       |
|    (Q)uit              *Case Insensitive*                     |
-----------------------------------------------------------------
choose one: d
================================

name:   Alice Zue
age:    20
bp:     110.00  76.00
risk:   0.0

name:   Bill Las
age:    21
bp:     130.00  83.00
risk:   2.0

name:   Cindy Sue
age:    33
```

```
bp:     134.00  90.70
risk:   4.0

name:   Dusan Luc
age:    60
bp:     168.50  130.20
risk:   5.5

========== 4 records ==========

----------------------------------------------------------------
|    (N)ew record        (R)emove record     Swa(p) records     |
|    (S)ort database     (C)lear database    (D)isplay db       |
|    (Q)uit              *Case Insensitive*                     |
----------------------------------------------------------------
choose one: p


----------------------------------------------------------------
|    (N)ew record        (R)emove record     Swa(p) records     |
|    (S)ort database     (C)lear database    (D)isplay db       |
|    (Q)uit              *Case Insensitive*                     |
----------------------------------------------------------------
choose one: d
==============================

name:   Bill Las
age:    21
bp:     130.00  83.00
risk:   2.0

name:   Alice Zue
age:    20
bp:     110.00  76.00
risk:   0.0

name:   Dusan Luc
age:    60
bp:     168.50  130.20
risk:   5.5

name:   Cindy Sue
age:    33
bp:     134.00  90.70
risk:   4.0

========== 4 records ==========

----------------------------------------------------------------
|    (N)ew record        (R)emove record     Swa(p) records     |
|    (S)ort database     (C)lear database    (D)isplay db       |
|    (Q)uit              *Case Insensitive*                     |
----------------------------------------------------------------
choose one:
```

Implement function `void swap(struct db_type * pArr[])` to accomplish swapping

Note that efficiency matters. In implementing swap, you should not use strcpy(), sprintf() etc to copy/move record data.

- When the user chooses `S` or `s`, which represents `Sort database`, the program further prompts the user with an option to sort records based on name (in alphabetic order) or risk factor (in ascending order). The function then sorts the database accordingly.

```
-----------------------------------------------------------------
|    (N)ew record        (R)emove record     Swa(p) records     |
|    (S)ort database     (C)lear database    (D)isplay db       |
|    (Q)uit              *Case Insensitive*                     |
-----------------------------------------------------------------
choose one: d
==============================

name:   Tim Kim
age:    25
bp:     130.00  76.00
risk:   1.0

name:   Alice Zue
age:    20
bp:     134.00  90.70
risk:   4.0

name:   Hysoon Pak
age:    60
bp:     168.50  130.20
risk:   5.5

name:   Cindy Sue
age:    33
bp:     110.00  75.6
risk:   0.0

========== 4 records ==========

-----------------------------------------------------------------
|    (N)ew record        (R)emove record     Swa(p) records     |
|    (S)ort database     (C)lear database    (D)isplay db       |
|    (Q)uit              *Case Insensitive*                     |
-----------------------------------------------------------------
choose one: s
sort by name (n) or risk (r)? n

-----------------------------------------------------------------
|    (N)ew record        (R)emove record     Swa(p) records     |
|    (S)ort database     (C)lear database    (D)isplay db       |
|    (Q)uit              *Case Insensitive*                     |
-----------------------------------------------------------------
choose one: d
==============================

name:   Alice Zue
age:    20
bp:     134.00  90.70
risk:   4.0
```

```
name:   Cindy Sue
age:    33
bp:     110.00  75.60
risk:   0.0

name:   Hysoon Pak
age:    60
bp:     168.50  130.20
risk:   5.5

name:   Tim Kim
age:    25
bp:     130.00  76.00
risk:   1.0

========== 4 records ==========

-----------------------------------------------------------------
|    (N)ew record        (R)emove record     Swa(p) records     |
|    (S)ort database     (C)lear database    (D)isplay db       |
|    (Q)uit              *Case Insensitive*                     |
-----------------------------------------------------------------
choose one: S
sort by name (n) or risk (r)? r

-----------------------------------------------------------------
|    (N)ew record        (R)emove record     Swa(p) records     |
|    (S)ort database     (C)lear database    (D)isplay db       |
|    (Q)uit              *Case Insensitive*                     |
-----------------------------------------------------------------
choose one: d
==============================

name:   Cindy Sue
age:    33
bp:     110.00  75.60
risk:   0.0

name:   Tim Kim
age:    25
bp:     130.00  76.00
risk:   1.0

name:   Alice Zue
age:    20
bp:     134.00  90.70
risk:   4.0

name:   Hysoon Pak
age:    60
bp:     168.50  130.20
risk:   5.5

========== 4 records ==========

-----------------------------------------------------------------
|    (N)ew record        (R)emove record     Swa(p) records     |
|    (S)ort database     (C)lear database    (D)isplay db       |
```

```
|     (Q)uit              *Case Insensitive*                      |
-------------------------------------------------------------------
choose one: q
sh-4.2$
```

Implement function `void sort(struct db_type * pArr[])` **to accomplish the sorting**

You can implement a sorting algorithm on your own, or explore and use a library function to do the sorting.
If you implement on your own, you should not use strcpy() sprintf() etc to copy/move record data.

Notes

- You should not add global or static variables in the program. Should not change main() method.
- Except displayDB(), all other functions that you implemented will alter the database.
- Assume input values (e.g., numerical values for age and blood pressures) are valid
- An executable file named **`sampleDB.out`**, which is my implementation for the assignment, is provided for you to try out different functionalities and their expected outputs. When running this file the first time, you may get "*Permission denied*" error. The reason is that the file does not have execute permission. Then issue command **`chmod 700 sampleDB.out`** to fix this.
  This file is generated from **`gcc`**, so run it the same way you run `a.out`
  Note that this file may not be run on some system (like some MAC terminal). Run it under the lab environment.
- Since the provided code for a2DB.c uses `fgets` to read user input. If you use `scanf` in your functions, you may experience some unexpected behaviors -- mixing `scanf` and `fget` has some problems. Thus in your implementation you should try to use `fgets` too. If you really want to use `scanf`, you may need to add a `getchar`() call after `scanf` to consume the extra chars leftover by `scanf`.

Submit your program by issuing submit command **`submit 2031AC a2 a2DB.c`**

or, use web-submit at **https://webapp.eecs.yorku.ca/submit/**

(see end of this pdf for more info)

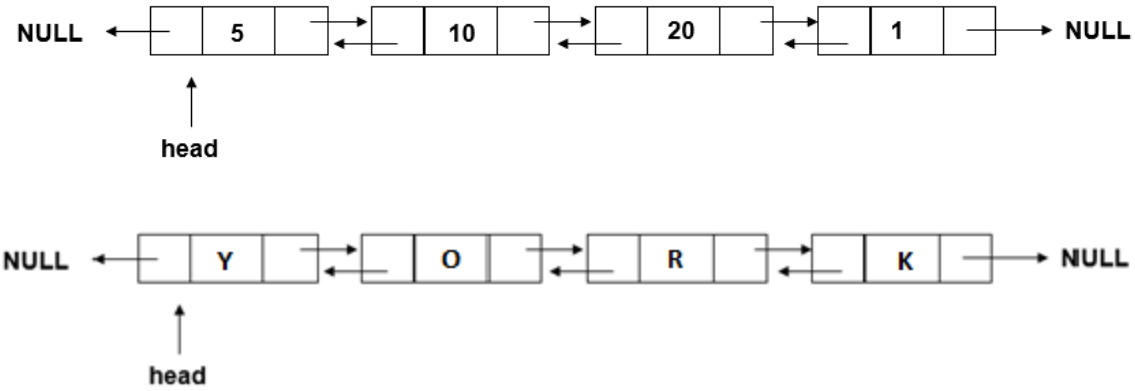
Problem 2 Doubly linked list in C (60 pts)

Subject: Stream IO, Structures, Array of structures, memory allocation

Specification

You have practiced in lab 6 the linked list where each node has one pointer to the next node. That kind of linked list is also called singly linked list. Based on that experience, here you write an ANSI-C program to implement a full-fledged doubly linked list data structure. In a doubly linked list, each node has two

pointers, one point to the next node in the chain and one points to the node preceding it. The first node's pointer to previous node is NULL, as shown below.

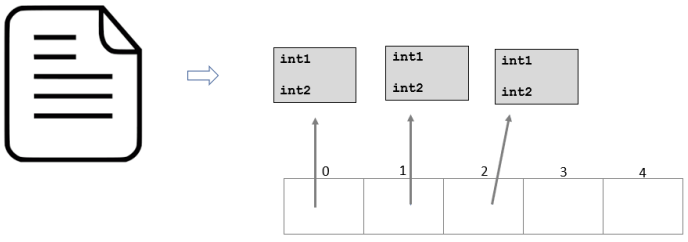


A doubly linked list can be traversed in both directions. At the cost of more space per node (due to the extra pointer to the preceding node), some operations can be more efficient than on a singly linked list.

Implementation

You are provided with a partially implemented program `a2dList.c`, and a data file `data.txt`. Study the existing code of the program, which does the following:

- Opens a data file `data.txt` using FILE IO in C. The file contains lines of integers, each line contains exactly two integers. (Stream and FILE IO is a topic that is usually in the syllabus but is skipped this semester.)
- Reads the data file line by line, and store the two integers in each line into a structure of type `twoInts`. The structures are maintained by `arr`, which is an array of pointers to struct `twoInts`. The structure `twoInts` contains two integer data members.
  - the structure pointed by the pointer in `arr[i]` gets the two values for its data members from the two integers in the `i`'th line of the file. For example, the structure pointed in `arr[0]` gets the two values for its members from the two integers in the first line of the file, `arr[1]` gets the two values for its members from the two integers in the second line, and so on.



Based on the existing implementation, implement the following:

- In main, build the linked list (pointed by `head`) by reading in pointee structures maintained in the pointer array, adding up the two integer fields and then inserting the sum value (as a char) into the linked list.
- Implement or complete the following functions.
  - `int len()`, which returns the number of nodes in the list.

- `char get(int index)`, which returns the data value of the node at index `index`. Assume the list is not empty, and `index` is in the range of `[0,len()-1]`.
- `int search(char key)` which searches the list for node with data `key`.
- `void insert(char c)`, which inserts at the end of the list a new node with data `c`. The list may or may not be empty.
- `void insertAfter(char c, int index)`, which inserts into the list a new node with data `c`, after the node at index `index`.
  Assume that the list is not empty, and `index` is in the range of `[0,len()-1]`.
- `void delete (char c)` which removes the node in the list that has data value `c`.
  Assume the node is not empty, and all the nodes in list have distinct data, and **the node with data `d` exists in the list**.

Hint: the slides and the Java program for lab6 will probably help you.
Don't add any global variables in your solution.

Sample Inputs/Outputs: (download – don't copy/paste - the input file `data.txt`)

```
red 419 % gcc a2dList.c
red 420 % a.out
arr[0]: 83 6
arr[1]: 71 8
arr[2]: 30 52
arr[3]: 26 49
arr[4]: 33 52
arr[5]: 40 5
arr[6]: 60 7
arr[7]: 2 81
arr[8]: 40 29

insert Y: (1)  Y    <-->    Y
insert O: (2)  Y  O    <-->    O  Y
insert R: (3)  Y  O  R    <-->    R  O  Y
insert K: (4)  Y  O  R  K    <-->    K  R  O  Y
insert U: (5)  Y  O  R  K  U    <-->    U  K  R  O  Y
insert -: (6)  Y  O  R  K  U  -    <-->    -  U  K  R  O  Y
insert C: (7)  Y  O  R  K  U  -  C    <-->    C  -  U  K  R  O  Y
insert S: (8)  Y  O  R  K  U  -  C  S    <-->    S  C  -  U  K  R  O  Y
insert E: (9)  Y  O  R  K  U  -  C  S  E    <-->    E  S  C  -  U  K  R  O  Y
remove S: (8)  Y  O  R  K  U  -  C  E    <-->    E  C  -  U  K  R  O  Y
remove E: (7)  Y  O  R  K  U  -  C    <-->    C  -  U  K  R  O  Y
remove -: (6)  Y  O  R  K  U  C    <-->    C  U  K  R  O  Y
remove O: (5)  Y  R  K  U  C    <-->    C  U  K  R  Y
remove R: (4)  Y  K  U  C    <-->    C  U  K  Y
remove K: (3)  Y  U  C    <-->    C  U  Y
remove Y: (2)  U  C    <-->    C  U
remove U: (1)  C    <-->    C
remove C: (0)
insert Y: (1)  Y    <-->    Y
insert O: (2)  Y  O    <-->    O  Y
insert R: (3)  Y  O  R    <-->    R  O  Y
insert K: (4)  Y  O  R  K    <-->    K  R  O  Y
insert U: (5)  Y  O  R  K  U    <-->    U  K  R  O  Y
insert -: (6)  Y  O  R  K  U  -    <-->    -  U  K  R  O  Y
insert C: (7)  Y  O  R  K  U  -  C    <-->    C  -  U  K  R  O  Y
insert S: (8)  Y  O  R  K  U  -  C  S    <-->    S  C  -  U  K  R  O  Y
insert E: (9)  Y  O  R  K  U  -  C  S  E    <-->    E  S  C  -  U  K  R  O  Y
```

```
get(0): Y
get(2): R
get(3): K
get(6): C
get(7): S
get(8): E

insert x after index 2: (10)
  Y  O  R  x  K  U  -  C  S  E    <-->    E  S  C  -  U  K  x  R  O  Y
insert y after index 0: (11)
  Y  y  O  R  x  K  U  -  C  S  E    <-->    E  S  C  -  U  K  x  R  O  y  Y
insert z after index 6: (12)
  Y  y  O  R  x  K  U  z  -  C  S  E    <-->    E  S  C  -  z  U  K  x  R  O  y  Y

get(0): Y
get(2): O
get(3): R
get(6): U
get(7): z
get(8): -

search - ....  found
search o ....  not found
search r ....  not found
search k ....  not found
search U ....  found
search x ....  found
search y ....  found
search Z ....  not found
search A ....  not found
search y ....  found
red 421 %
```

Submit your program by issuing **`submit 2031AC a2 a2dList.c data.txt`**

or websubmit (see below for more info)

Make sure your program compiles in the lab environment. The program that does not compile in the lab environment will get 0.

In summary, for this assignment you should submit three files: **`a2DB.c a2dList.c data.txt`**

At any time and from any directory, you can issue **`submit -l 2031AC a2`** to view the list of files that you have submitted. Or check from websubmit page.

Lower case L

Common Notes

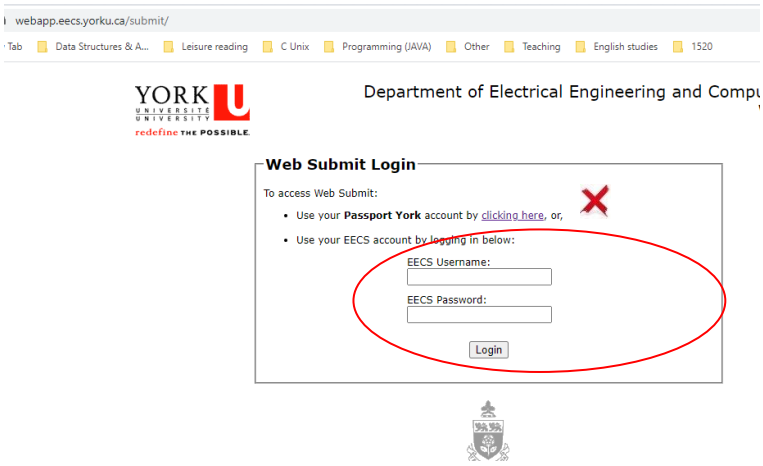
All submitted files should contain the following header:

```
/**************************************
* EECS2031 21F - Assigment 2 *
* Author: Last name, first name *
* Email: Your email address *
* eecs_username: Your EECS login username *
* York num: Your YorkU student number
**************************************/
```

Web-submit

To prepare for the coming labtest, for which you will be encouraged to work on your local computer and submit using websumit, for this assignment you can submit using websubmit. So, in addition to submitting from the lab, you can also submit from local machine directly. (You should still make sure that the files compile in the lab environment)
You can submit using websubmit at **https://webapp.eecs.yorku.ca/submit/**

- **login using EECS , not passport York. If you keep on getting prompted to login using passport York, then clear the cookie and cache of your browser and then try again. Eventually you should see the page below.**
- **Once login, select Course: 2031AC and then select Assignment: a2**
- **Then upload files from your local computer.**



Use EECS account

See this page for more information about web-submit
https://wiki.eecs.yorku.ca/dept/tdb/services:submit:websubmit