

# Assignment 2 Self-referential structures, structures, pointer arrays

Release: Mar 18

Due: Apr 6 (Thursday) 11:00 pm

Total marks: 120 pts

Note that unlike the weekly labs, this is an individual work, and you should not discuss with others. Ask the instructor for help.

In this assignment (also called Program Exam in some courses), you are going to implement a customized data structure and also implement a self-referential data structure (linked list).

## Problem 1 Linked list in C (40 pts)

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

### Specification

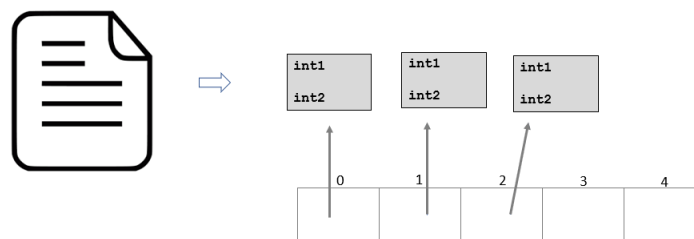
You have practiced in lab 6 the simple linked list where each node has one pointer to the next node. That kind of linked list is also called singly linked list. Based on that experience, here you write an ANSI-C program to implement a full-fledged linked list data structure.

### Implementation

You are provided with a partially implemented program `a2LList.c`, and a data file `data.txt`. You are also provided with an object file `functions.o`, which contains implementations of some functions on linked list, such as `len()`, `get()`, `print()`. (An object file is a compiled file that contains machine code. You cannot view the code in this file but you use this file by compiling together with your code, as shown in sample input/output.

Study the existing code of the c file, 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 stores 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 function, build the linked list (pointed by `head`) by reading in pointer 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 `c` 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 a2LList.c functions.o
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 16
arr[7]: 0 65
arr[8]: 2 81
```

```
insert Y: (1)    Y
insert O: (2)    Y    O
insert R: (3)    Y    O    R
insert K: (4)    Y    O    R    K
insert U: (5)    Y    O    R    K    U
insert -: (6)    Y    O    R    K    U    -
insert L: (7)    Y    O    R    K    U    -    L
insert A: (8)    Y    O    R    K    U    -    L    A
insert S: (9)    Y    O    R    K    U    -    L    A    S
remove S: (8)    Y    O    R    K    U    -    L    A
remove A: (7)    Y    O    R    K    U    -    L
remove -: (6)    Y    O    R    K    U    L
remove O: (5)    Y    R    K    U    L
remove R: (4)    Y    K    U    L
remove K: (3)    Y    U    L
remove Y: (2)    U    L
```

```

remove U: (1)    L
remove L: (0)
insert Y: (1)    Y
insert O: (2)    Y    O
insert R: (3)    Y    O    R
insert K: (4)    Y    O    R    K
insert U: (5)    Y    O    R    K    U
insert -: (6)    Y    O    R    K    U    -
insert L: (7)    Y    O    R    K    U    -    L
insert A: (8)    Y    O    R    K    U    -    L    A
insert S: (9)    Y    O    R    K    U    -    L    A    S

get(0): Y
get(2): R
get(3): K
get(6): L
get(7): A
get(8): S

insert x after index 2: (10)    Y    O    R    x    K    U    -    L    A    S
insert y after index 0: (11)    Y    y    O    R    x    K    U    -    L    A    S
insert z after index 6: (12)    Y    y    O    R    x    K    U    z    -    L    A    S

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 .... found
search y .... found
search @ .... not found
red 421 %

```

Submit all your program files by issuing `submit 2031M a2 a2LList.c data.txt function.o`

## Question 2 pointer arrays, structures in C (80 pts)

**Subject:** Structures, array of pointer to structures. 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.

**Specification:** in this exercise you are going to develop a simple database management system for a cruise company.

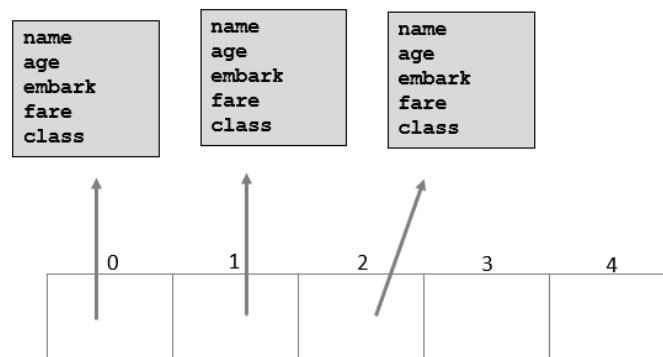
The system maintains records for its passengers for a trip. Each record contains name, age, embark port, and the fare paid, and the cabin class (e.g., Economic, Middle, UpperMiddle), which is based on the embark port and fare. If a passenger's fare info has been updated, then a history of classes is maintained.

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

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

- `name` of type `char []`, which represents the passenger's name
- `int age`, which represents the passenger's age
- `embark`, which represents the embark port of the passenger
- `float fare`, which represents the passenger's paid fare for the trip
- `class` of type `char []`, which represents the passenger's cabin class.

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 functionalities:

- Accepting entry of a new passenger record into the current database, and generating the cabin class for the passenger.
- Displaying all records in the current database
- Removing an existing passenger's record in the current database
- Updating fare info of an existing passenger, and generating new cabin class for the member
- Sorting the 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 N/n, D/d, U/u, C/c, R/r, S/s 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      (U)pdate record      |
|      (S)ort database   (C)lear database   (D)isplay database |
|      (Q)uit            *Case Insensitive*      |
|-----|
```

```
choose one: x
```

```
not a valid input!
```

```
-----
|      (N)ew record      (R)emove record      (U)pdate record      |
|      (S)ort database   (C)lear database   (D)isplay database |
|      (Q)uit            *Case Insensitive*      |
|-----|
```

```
choose one: sort
```

```
not a valid input!
```

```
-----
|      (N)ew record      (R)emove record      (U)pdate record      |
|      (S)ort database   (C)lear database   (D)isplay database |
|      (Q)uit            *Case Insensitive*      |
|-----|
```

```
choose one:
```

- When the users enter n or N, which represents New record, the program further prompts the user to enter a new passenger record into the database. The function reads in the passenger's name, age, embark port and fare paid (in \$). It then calculates the cabin class for the new passenger. The function then creates a new passenger record and inserts the new record into the database (i.e., the pointer array).

The company operates three embark ports: **Cherbourg, Queenstown, Southampton**

Cabin classification is calculated based on embark port and the fare paid, as show in the table:

Cherbourg	Queenstown	Southampton	Cabin classes
fare < 9	fare < 7	fare < 8	<i>Economic</i>
$9 \leq \text{fare} < 30$	$7 \leq \text{fare} < 18$	$8 \leq \text{fare} < 24$	<i>LowerMiddle</i>
$30 \leq \text{fare} < 100$	$18 \leq \text{fare} < 76$	$24 \leq \text{fare} < 80$	<i>Middle</i>
$100 \leq \text{fare} < 200$	$76 \leq \text{fare} < 150$	$80 \leq \text{fare} < 180$	<i>UpperMiddle</i>
fare $\geq 200$	fare $\geq 150$	fare $\geq 180$	<i>Wealthy</i>

Assume the user enters the correct port name, i.e., one of *Cherbourg*, or *Queenstown* or *Southampton*. No need to do error checking (but you are welcome to add that)

- 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      (U)pdate record      |
|      (S)ort database   (C)lear database   (D)isplay database |
|      (Q)uit            *Case Insensitive*   |
-----

```

choose one: **d**

```

=====
===== 0 records =====

```

```

-----
|      (N)ew record      (R)emove record      (U)pdate record      |
|      (S)ort database   (C)lear database   (D)isplay database |
|      (Q)uit            *Case Insensitive*   |
-----

```

choose one: **n**

name: **Judy Sue**

age: **30**

embark port: **Cherbourg**

fare (\$): **21.3**

```

-----
|      (N)ew record      (R)emove record      (U)pdate record      |
|      (S)ort database   (C)lear database   (D)isplay database |
|      (Q)uit            *Case Insensitive*   |
-----

```

choose one: **d**

```

=====

```

name: Judy Sue

age: 30

embark: Cherbourg

fare: 21.3

class: LowerMiddle

```

===== 1 records =====

```

```

-----
|      (N)ew record      (R)emove record      (U)pdate record      |
|      (S)ort database   (C)lear database   (D)isplay database |
|      (Q)uit            *Case Insensitive*   |
-----

```

choose one: **n**

name: **Cindy White**

age: **52**

embark port: **Queenstown**

fare (\$): **73**

```

-----
|      (N)ew record      (R)emove record      (U)pdate record      |
|      (S)ort database   (C)lear database   (D)isplay database |
|      (Q)uit            *Case Insensitive*   |
-----

```

choose one: **d**

```

=====

```

name: Judy Sue

age: 30

embark: Cherbourg

```
fare: 21.3
class: LowerMiddle
```

```
name: Cindy White
age: 52
embark: Queenstown
fare: 73.0
class: Middle
```

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

```
-----
|      (N)ew record      (R)emove record      (U)pdate record      |
|      (S)ort database   (C)lear database   (D)isplay database |
|      (Q)uit            *Case Insensitive*    |
-----
```

choose one: **c**

are you sure to clear database? (y) or (n)? **y**

```
-----
|      (N)ew record      (R)emove record      (U)pdate record      |
|      (S)ort database   (C)lear database   (D)isplay database |
|      (Q)uit            *Case Insensitive*    |
-----
```

choose one: **d**

=====

===== 0 records =====

```
-----
|      (N)ew record      (R)emove record      (U)pdate record      |
|      (S)ort database   (C)lear database   (D)isplay database |
|      (Q)uit            *Case Insensitive*    |
-----
```

choose one:

**Implement/complete functions `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, the system should further prompt the user for the name of the member 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 like in the assignment 1, after removal, the relative ordering of the remaining records should remain unchanged, as shown below.**

```
-----
|      (N)ew record      (R)emove record      (U)pdate record      |
|      (S)ort database   (C)lear database   (D)isplay database |
|      (Q)uit            *Case Insensitive*    |
-----
```

choose one: **d**

```
name: Alice Zue
age: 20
embark: Cherbourg
fare: 18.1
class: LowerMiddle
```

name: Bill Las  
age: 21  
embark: Queenstown  
fare: 78.0  
class: UpperMiddle

name: Cindy White  
age: 52  
embark: Queenstown  
fare: 73.0  
class: Middle

name: Dusan Luc  
age: 60  
embark: Southampton  
fare: 79.8  
class: Middle

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

```
-----  
|      (N)ew record      (R)emove record      (U)pdate record      |  
|      (S)ort database   (C)lear database   (D)isplay database   |  
|      (Q)uit            *Case Insensitive*      |  
-----
```

choose one: **r**  
enter full name to remove: **Linh Ng**  
record not found!

```
-----  
|      (N)ew record      (R)emove record      (U)pdate record      |  
|      (S)ort database   (C)lear database   (D)isplay database   |  
|      (Q)uit            *Case Insensitive*      |  
-----
```

choose one: **r**  
enter full name to remove: **Bill las**  
record not found!

```
-----  
|      (N)ew record      (R)emove record      (U)pdate record      |  
|      (S)ort database   (C)lear database   (D)isplay database   |  
|      (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      (U)pdate record      |  
|      (S)ort database   (C)lear database   (D)isplay database   |  
|      (Q)uit            *Case Insensitive*      |  
-----
```

choose one: **d**

=====

name: Alice Zue  
age: 20  
embark: Cherbourg



fare: 18.1  
class: LowerMiddle

name: Cindy White  
age: 52  
embark: Queenstown  
fare: 73.0  
class: Middle

name: Dusan Luc  
age: 60  
embark: Southampton  
fare: 79.8  
class: Middle

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

```
-----  
|      (N)ew record      (R)emove record      (U)pdate record      |  
|      (S)ort database    (C)lear database    (D)isplay database    |  
|      (Q)uit             *Case Insensitive*      |  
-----
```

choose one: **r**  
enter full name to remove: **Cindy**  
record not found!

```
-----  
|      (N)ew record      (R)emove record      (U)pdate record      |  
|      (S)ort database    (C)lear database    (D)isplay database    |  
|      (Q)uit             *Case Insensitive*      |  
-----
```

choose one: **R**  
enter full name to remove: **Cindy White**  
record [Cindy White] removed successfully.

```
-----  
|      (N)ew record      (R)emove record      (U)pdate record      |  
|      (S)ort database    (C)lear database    (D)isplay database    |  
|      (Q)uit             *Case Insensitive*      |  
-----
```

choose one: **d**

=====

name: Alice Zue  
age: 20  
embark: Cherbourg  
fare: 18.1  
class: LowerMiddle

name: Dusan Luc  
age: 60  
embark: Southampton  
fare: 79.8  
class: Middle

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

-----

(N)ew record	(R)emove record	(U)pdate record
(S)ort database	(C)lear database	(D)isplay database
(Q)uit	*Case Insensitive*	

choose one:

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

- When the user chooses `u` or `U`, which represents `Update record`, the system allows a user to update fare info of existing passengers. The system prompts the user for the name of the passenger whose record is to be update (case sensitive). If no record by that name is found in the current database, then the error message *"record not found!"* should be displayed. If the record is found, the system further prompts the user for a new fare. It then displays message *"record [xx] updated successfully"*. Internally, the system replaces the existing fare of the passenger. Also, calculate the new cabin class using the new fare, and append the new class to the existing class(es). (Assume a passenger's embark port never changes). This way, the system keeps track of the class changes of the passenger.

(N)ew record	(R)emove record	(U)pdate record
(S)ort database	(C)lear database	(D)isplay database
(Q)uit	*Case Insensitive*	

choose one: **d**

```
=====
name:  Alice Zue
age:   20
embark: Cherbourg
fare:  18.1
class: LowerMiddle
```

```
name:  Dusan Luc
age:   60
embark: Southampton
fare:  79.8
class: Middle
```

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

(N)ew record	(R)emove record	(U)pdate record
(S)ort database	(C)lear database	(D)isplay database
(Q)uit	*Case Insensitive*	

choose one: **u**

enter full name to search: **Alice**

record not found!

(N)ew record	(R)emove record	(U)pdate record
(S)ort database	(C)lear database	(D)isplay database
(Q)uit	*Case Insensitive*	

choose one: **U**

enter full name to search: **Alice Zue**

record found, enter new fare (\$): **78.6**  
record [Alice Zue] updated successfully.

```
-----  
|      (N)ew record      (R)emove record      (U)pdate record      |  
|      (S)ort database   (C)lear database   (D)isplay database   |  
|      (Q)uit            *Case Insensitive*      |  
-----
```

choose one: **d**

=====

name: Alice Zue  
age: 20  
embark: Cherbourg  
fare: 78.6  
class: LowerMiddle -> Middle

name: Dusan Luc  
age: 60  
embark: Southampton  
fare: 79.8  
class: Middle

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

```
-----  
|      (N)ew record      (R)emove record      (U)pdate record      |  
|      (S)ort database   (C)lear database   (D)isplay database   |  
|      (Q)uit            *Case Insensitive*      |  
-----
```

choose one: **u**

enter full name to search: **Dusan Luc**  
record found, enter new fare (\$): **190**  
record [Dusan Luc] updated successfully.

```
-----  
|      (N)ew record      (R)emove record      (U)pdate record      |  
|      (S)ort database   (C)lear database   (D)isplay database   |  
|      (Q)uit            *Case Insensitive*      |  
-----
```

choose one: **d**

=====

name: Alice Zue  
age: 20  
embark: Cherbourg  
fare: 78.6  
class: LowerMiddle -> Middle

name: Dusan Luc  
age: 60  
embark: Southampton  
fare: 190.0  
class: Middle -> Wealthy

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

-----

(N)ew record	(R)emove record	(U)pdate record
(S)ort database	(C)lear database	(D)isplay database
(Q)uit	*Case Insensitive*	

```

choose one: U
enter full name to search: Alice Zue
record found, enter new fare ($): 102
record [Cindy White] updated successfully.

```

(N)ew record	(R)emove record	(U)pdate record
(S)ort database	(C)lear database	(D)isplay database
(Q)uit	*Case Insensitive*	

```

choose one: U
enter full name to search: Dusan Luc
record found, enter new fare ($): 160.1
record [Dusan Luc] updated successfully.

```

(N)ew record	(R)emove record	(U)pdate record
(S)ort database	(C)lear database	(D)isplay database
(Q)uit	*Case Insensitive*	

```

choose one: d

```

```

=====

```

```

sname:  Alice Zue
age:    20
embark: Cherbourg
fare:   102.0
class:  LowerMiddle -> Middle -> UpperMiddle

```

```

name:    Dusan Luc
age:     60
embark:  Southampton
fare:    160.1
class:   Middle -> Wealthy -> UpperMiddle

```

```

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

```

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

- 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 based on the current fare (in ascending order). The function then sorts the database accordingly. Assume the user entered either n or f (so you don't need to do error checking)

(N)ew record	(R)emove record	(U)pdate record
(S)ort database	(C)lear database	(D)isplay database
(Q)uit	*Case Insensitive*	

```

choose one: d

```

```

=====

```

name: Alice Zue  
age: 20  
embark: Cherbourg  
fare: 102.0  
class: LowerMiddle -> Middle -> UpperMiddle

name: Dusan Luc  
age: 60  
embark: Southampton  
fare: 160.1  
class: Middle -> Wealthy -> UpperMiddle

name: Tina Sue  
age: 20  
embark: Queenstown  
fare: 6.3  
class: Economic

name: Bill Las  
age: 21  
embark: Cherbourg  
fare: 24.0  
class: LowerMiddle

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

```
-----  
|      (N)ew record      (R)emove record      (U)pdate record      |  
|      (S)ort database   (C)lear database   (D)isplay database |  
|      (Q)uit            *Case Insensitive*      |  
-----
```

choose one: **s**  
sort by name (n) or fare (f)? **n**

```
-----  
|      (N)ew record      (R)emove record      (U)pdate record      |  
|      (S)ort database   (C)lear database   (D)isplay database |  
|      (Q)uit            *Case Insensitive*      |  
-----
```

choose one: **d**

=====

name: Alice Zue  
age: 20  
embark: Cherbourg  
fare: 102.0  
class: LowerMiddle -> Middle -> UpperMiddle

name: Bill Las  
age: 21  
embark: Cherbourg  
fare: 24.0  
class: LowerMiddle

name: Dusan Luc  
age: 60  
embark: Southampton

fare: 160.1  
class: Middle -> Wealthy -> UpperMiddle

name: Tina Sue  
age: 20  
embark: Queenstown  
fare: 6.3  
class: Economic

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

(N)ew record	(R)emove record	(U)pdate record
(S)ort database	(C)lear database	(D)isplay database
(Q)uit	*Case Insensitive*	

choose one: **S**

sort by name (n) or fare (f)? **f**

(N)ew record	(R)emove record	(U)pdate record
(S)ort database	(C)lear database	(D)isplay database
(Q)uit	*Case Insensitive*	

choose one: **d**

name: Tina Sue  
age: 20  
embark: Queenstown  
fare: 6.3  
class: Economic

name: Bill Las  
age: 21  
embark: Cherbourg  
fare: 24.0  
class: LowerMiddle

name: Alice Zue  
age: 20  
embark: Cherbourg  
fare: 102.0  
class: LowerMiddle -> Middle -> UpperMiddle

name: Dusan Luc  
age: 60  
embark: Southampton  
fare: 160.1  
class: Middle -> Wealthy -> UpperMiddle

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

(N)ew record	(R)emove record	(U)pdate record
(S)ort database	(C)lear database	(D)isplay database
(Q)uit	*Case Insensitive*	

```
choose one: q
red 327 %
```

Implement function `void sortDB(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 fare, embark port) are valid, so no error checking is needed
- 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 for 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 is compiled under the lab environment, and may not run on some other system (like some MAC terminal). Run it under the lab environment.

- Since the provided code in `a2DB.c` uses `fgets` to read user input. If you use `scanf` in your functions, you may experience some unexpected behaviors -- mixing `scanf` and `fgetc` has some problems because there may be extra chars left over by `scanf`. 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 each `scanf` to consume the extra chars leftover by `scanf`.

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

---

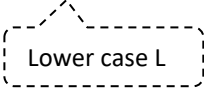
Make sure your program compiles in the lab environment. The program that does not compile in the lab environment will get 0.

In summary, for this assignment you should submit four files:

`a2IList.c`    `data.txt`    `functions.o`

`a2DB.c`

At any time and from any directory, you can issue `submit -l 2031M 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:

```
/*****  
* 23W - Programming Assignment 2 *  
* Author: Last name, first name *  
* EECS/Prism username: Your eecs login username *  
* Yorku Student #: Your student number *  
* Email: Your email address *  
*****/
```

## Other common notes:

- Make sure your program compiles in the lab environment. The program that does not compile, or, crashes with “segmentation fault” in the lab will get 0.
- Note that programming assignments are individual work. Unlike labs, you should NOT discuss with others. Doing so is considered a violation of academic honesty.
- Note that submitting previous term's files – even it is yours -- is considered self-plagiarism and thus will receive 0.
- All submissions need to be done from the lab, using command line.
  - Also note that you can submit the same file multiple times. Then the latest file will overwrite the old one.
  - If you submitted a wrong file, you cannot delete it. Ask the instructor to delete it for you.