Files

Storage Room

Workshop 9 (worth 3% of your final grade)
URL: https://github.com/Seneca-144100/IPC-WS9

In this workshop, you are to write an application that keeps a saved inventory of labelled boxes in a storage room. In this application, you can list the boxes in the storage, or query the system for the specifications of a certain box. You can also add information about a new box to the saved file. To make it more fun, the application can pick up a lucky box!

LEARNING OUTCOMES

Upon successful completion of this workshop, you will have demonstrated the abilities

- Open and close a text file
- Read sequentially from a text file
- Append to a text file
- Get a random integer

SUBMISSION POLICY

Your workshops are divided in two sections; in_lab and at_home.

The "in_lab" section is to be completed **during your assigned lab section**. It is to be completed and submitted by the end of the workshop. If you do not attend the workshop, you can submit the "in_lab" section along with your "at_home" section (a 20% late deduction will be assessed). The "at_home" portion of the lab is **due the day before your next scheduled workshop**

All your work (all the files you create or modify) must contain your name, Seneca email and student number.

You are responsible for regularly backing up your work.

IN-LAB: READ AND SEARCH A TEXT FILE (70%)

Download or clone workshop 9 from https://github.com/Seneca-144100/IPC-WS9

The structure used for the data regarding a box is:

```
struct Box{
      int id;                 // the box ID
      double size[3];  // dimensions of the box (Length, Width, Height)
      double weight;   // weight of the box
};
```

1. Code the function *listBoxes* that inputs a file name and displays all the boxes saved in that file. See the sample output.

```
void listBoxes(const char filename[]);
```

Instructions:

- Open the file in read mode.
- If unable to open the file, print a message.
- In a loop, read a record and print using `"%2d %6.2lf %5.2lf %6.2lf %6.2lf\n"` format. See the sample output.
- Close the file.

2. Call *listBoxes* in the *main* function when menu option 1 has been selected.

3. Code the function *searchBox* that searches for a box given a file name and a box ID, and returns the record number if found.

```
int searchBox(FILE *fp, int id2Find)
```

Instructions:

- Return -1 if fp is NULL; otherwise,
- Rewind to go to the beginning of the file.
- Loop through the file, reading one record at a time. If the ID of the record matches the input ID, return the record number.
- If a matching ID is not found, return -1.

4. Code the function *displayBox* that displays the data regarding a box given the box's ID.

```
void displayBox(const char filename[], int id2Find);
```

Instructions:

- Open the file in read mode, prompt if not successful.

- Call *searchBox* to find the record number and print appropriate message.
- If found,
    - Rewind to go to the beginning of the file.
    - Loop through the file to get to the proper box.
    - Call *printBox* (already implemented for you) to output the details.
- Close the file.

5. Complete code in the *main* function for the case in which menu option 2 has been selected. Ask the user to enter a box ID. Then call *displayBox* to show its details.

Output Sample:

```
Welcome to My Storage Room
==========================
1- List all boxes
2- Find a box
3- Add a box
4- Randomly pick a lucky box!
0- Exit program
Select an option: 1
List of boxes
=============

ID Length Width Height Weight
-----------------------------
10  50.34 61.00  30.00  50.50
11  25.60 12.34   9.23  12.89
55  10.00 20.00  30.00  40.50
56  30.00 40.00  50.00  60.00

1- List all boxes
2- Find a box
3- Add a box
4- Randomly pick a lucky box!
0- Exit program
Select an option: 2
Enter box ID: 55
Found box 55 as record #3:

ID:         55
Length:  10.00
Width:   20.00
Height:  30.00
Weight:  40.50
```

```
1- List all boxes
2- Find a box
3- Add a box
4- Randomly pick a lucky box!
0- Exit program
Select an option: 2
Enter box ID: 15
This box is not recorded.

1- List all boxes
2- Find a box
3- Add a box
4- Randomly pick a lucky box!
0- Exit program
Select an option: 0
```

For submission instructions, see the SUBMISSION section below.

IN_LAB SUBMISSION:

To test and demonstrate execution of your program use the same data as the output example above or any information needed….

If not on matrix already, upload your **w9_in_lab.c** to your matrix account. Compile and run your code and make sure everything works properly.

Then run the following script from your account:

~profname.proflastname/submit ipc_w9_in_lab <ENTER>

and follow the instructions.

AT_HOME: WRITE IN A TEXT FILE (20%)

After completing the in_lab section, copy w9_in_lab.c to w9_at_home.c and upgrade the code using the following instructions.

1. Code the function *addBox* that adds a box to the file, if a box with the same ID does not exist in the file, and returns the number of boxes added (0 or 1).

```
int addBox(const char filename[], const struct Box * b2Add);
```

Instructions:

1. Open the file in “a+” mode, print a message if not successful.
2. Call searchBox to check if a box with ID *b2Add* already exists in the file or not.
3. If found, print a message (see sample output); otherwise,
4. Write the box details to the file. Use format: `"%d %.2lf %.2lf %.2lf %.2lf\n"`
5. Close the file

2. Complete code in the *main* function for the case in which menu option 3 has been selected. Ask the user to enter details about a box (see sample output). Then call *addBox* to add the box to the file. Print a message confirming how many boxes were added to the file.

3. Code the function *getRandomInt* that generates a random integer between a lower and higher value.

```
int getRandomInt(int lower, int higher);
```

Note: make sure that you set the seed before calling the rand() function

4. Code the function *numberBoxes* that returns the number of boxes saved in a file.

```
int numberBoxes(const char filename[]);
```

5. Code the function *displayBoxN* that displays the details of the N^{th} record in the file.

Instructions:

1. Open the file in read mode, print a message if not successful.
2. Read through the file to get to the N^{th} record.
3. Call *printBox* to print box details.
4. Close the file.

6. Complete code in the *main* function for the case in which menu option 4 has been selected. Use the time function to set the seed for the random generator. Then call *getRandomInt* to generate a number, n, between 1 and the number of records in the file (use *numberBoxes)*. Then call *displayBoxN* to display the details of record #n.

Output Sample:

```
Welcome to My Storage Room
==========================
1- List all boxes
2- Find a box
3- Add a box
4- Randomly pick a lucky box!
0- Exit program
Select an option: 1
List of boxes
=============

ID Length Width Height Weight
-----------------------------
10  50.34 61.00  30.00  50.50
11  25.60 12.34   9.23  12.89
55  10.00 20.00  30.00  40.50
56  30.00 40.00  50.00  60.00

1- List all boxes
2- Find a box
3- Add a box
4- Randomly pick a lucky box!
0- Exit program
Select an option: 3
Please enter the box's ID, length, width, height and weight: 55 5.2
6.3 7.4 8.5
A box with this ID is already recorded.
0 box added to storage!

1- List all boxes
2- Find a box
3- Add a box
4- Randomly pick a lucky box!
```

```
0- Exit program
Select an option: 3
Please enter the box's ID, length, width, height and weight: 57 5.2
6.3 7.4 8.5
1 box added to storage!

1- List all boxes
2- Find a box
3- Add a box
4- Randomly pick a lucky box!
0- Exit program
Select an option: 1
List of boxes
=============

ID Length Width Height Weight
-----------------------------
10  50.34 61.00  30.00  50.50
11  25.60 12.34   9.23  12.89
55  10.00 20.00  30.00  40.50
56  30.00 40.00  50.00  60.00
57   5.20  6.30   7.40   8.50

1- List all boxes
2- Find a box
3- Add a box
4- Randomly pick a lucky box!
0- Exit program
Select an option: 4

Lucky box picked:

ID:         56
Length:  30.00
Width:   40.00
Height:  50.00
Weight:  60.00

1- List all boxes
2- Find a box
3- Add a box
4- Randomly pick a lucky box!
0- Exit program
Select an option: 4
```

```
Lucky box picked:

ID:         11
Length:  25.60
Width:   12.34
Height:   9.23
Weight:  12.89

1- List all boxes
2- Find a box
3- Add a box
4- Randomly pick a lucky box!
0- Exit program
Select an option: 0
```

AT-HOME REFLECTION (10%)

Please provide brief answers to the following questions in a text file named **`reflect.txt`**.

1) What is the difference between opening a file in “a” mode versus “a+” mode?
2) In this workshop, a file was used to save a table. What was the first record in this table? What was the third field?
3) Why do we need to call the function srand()? What happens if we don’t call it before using rand()?

AT_HOME SUBMISSION:

To test and demonstrate execution of your program use the same data as the output example above.

If not on matrix already, upload your **`reflect.txt`** and **`w9_at_home.c`** to your matrix account. Compile and run your code and make sure everything works properly.

***Important:* Comment out the line that sets the seed for random number generation**

Then run the following script from your account:

`~profname.proflastname/submit ipc_w9_at_home <ENTER>`

and follow the instructions.