COMP1511 18s2

Week-08 Laboratory Exercises

Topics

Preparation

Before the lab you should re-read the relevant lecture slides and their accompanying examples. **We will only assess lab Exercises 01, 02, 03 and 04 to derive marks for this lab.** However, you will learn a lot by attempting and solving the challenge exercises.

Getting Started

Create a new directory for this lab called `lab08` by typing:

```
$ mkdir lab08
```

Change to this directory by typing:

```
$ cd lab08
```

Exercise-00: Reminder - Mandelbrot Art Gallery (individual) (same as in week-07)

This is an individual exercise to complete by yourself.

This exercise is due at the end of **week 8** (two weeks from now)

Check out the **MandelbrArt Gallery!**

You've created your Mandelbrot server, you've got the technical side of things working... now it's time to actually **use** your code, and to exercise your aesethetic senses.

Find/create the most beautiful colored tile from the Mandelbrot set that you can. Submit your pixelColor.c file from the Mandelbrot activity AND also the file xanadu.h (sample below) which defines the x y and z coordinates for your tile.

```
/*
 * xanadu.h
 * Defines the centre of your beautiful colored tile image of the
 * Mandelbrot set.
 * Your name here:
 * Your description here:
 */

// Replace these three values with your own chosen numbers
#define CENTER_X 1.0
#define CENTER_Y 0.0
#define ZOOM 9
```

We'll regenerate the tile using your code, and display them in the MandelbrArt Gallery.

Note that you can submit as many times as you want, but only your latest submission will be displayed in the gallery (so don't be afraid to experiment!).

Don't forget to also submit your pixelColor.c, so that we can re-generate the tile using your color scheme.

When you are finished working on this exercise you must submit your work by running **give**:

```
$ give cs1511 wk07_mandelbrotBeauty xanadu.h pixelColor.c
```

You must run give before **Sunday 16 September 23:59:59** (end of **week 8**) to obtain the marks for this lab exercise. Note, this is an individual exercise, the work you submit with **give** must be entirely your own.

Exercise-01: How many Orca Pods in a List (pair)

This is a pair exercise to complete with your lab partner.

This week's lab exercises will use the same data as last week but with different data structures. First, a reminder of the data.
A citizen science project monitoring whale populations has files containing large numbers of whale observations. Each line in these files contains:

- the date the observation was made
- the number of whales in the pod
- the species of whales

Here is a file of example data which you can download to test your program. It looks like this:

```
$ head whales.txt
18/01/18  9 Pygmy right whale
01/09/17 21 Southern right whale
16/02/18  4 Striped dolphin
02/07/17  4 Common dolphin
19/02/18  4 Blue whale
21/02/18 38 Dwarf sperm whale
14/06/17 29 Southern right whale
20/06/17  3 Spinner dolphin
22/07/17 34 Indo-Pacific humpbacked dolphin
20/03/18  7 Long-finned pilot whale
```

Here is the starting code for this exercise.
It already contains the functions which you discussed in your tutorial which reads the file of whale sightings into a linked list of structs. You do not need need to change these functions.

As you discussed in your tutorial, the struct used to represent each whale sighting has changed from last week. It is now this:

```
struct pod {
    struct pod  *next;
    struct date *when;
    int         how_many;
    char        *species;
};
```

The main function in **orca_list.c** has this code:

```
int n_orca_pods = count_orca_sightings(first_pod);
printf("%d Orca sightings in %s\n", n_orca_pods, argv[1]);
```

Your task in this exercise is to complete this function:

```
int count_orca_sightings(struct pod *first_pod) {
    // REPLACE THIS COMMENT WITH YOUR CODE
    // YOU ARE NOT PERMITTED TO USE ARRAYS
    // THIS FUNCTION SHOULD NOT CALL SCANF OR PRINTF
    // IT SHOULD JUST RETURN A VALUE
    return 42; // CHANGE ME
}
```

Do **not** change any other function.

You are not permitted to use an array in your answer to this question.

It should return a count of the number of sightings of Orca pods in the file.

Note, "pod" is the collective number for a group of whales. Your functions should return the number of Orca pods. It does not have have to sum the number of individual whales in these pods.

When you have completed the function **count_orca_sightings** this is how **orca_list.c** should behave:

```
$ dcc -o orca_list orca_list.c
$ ./orca_list whales.txt
53 Orca sightings in whales.txt
```

Hint: only small changes are necessary to last week's code

Hint: if you need help understanding how structs can be linked into a list with pointers watch this great explanation from COMP1511 tutor Mark Sonnenschein:

Hint: if you you need to revise pointers or structs these videos prepared by COMP1511 tutor may help:

- Structs - the Basics
- Pointers - why use them
- Pointers - Walkthrough (Coding)
- Pointers - Walkthrough using a swap function
- Pointers - Walkthrough using a swap function (coding)

When you think your program is working you can use `autotest` to run some simple automated tests:

```
$ 1511 autotest orca_list
```

When you are finished on this exercise you and your lab partner must both submit your work by running **give**:

```
$ give cs1511 wk08_orca_list orca_list.c
```

Note, even though this is a pair exercise, you both must run **give** from your own account before **Sunday 16 September 23:59:59** to obtain the marks for this lab exercise.

Exercise-02: Counting A Whale Species in a List (pair)

This is a pair exercise to complete with your lab partner.

Here is the starting code for this exercise.
It already contains the functions which you discussed in your tutorial which reads the file of whale sightings into a linked list of structs. You do not need need to change these functions.

The main function in **species_count_list.c** has this code:

```
struct pod *first_pod = read_sightings_file(argv[1]);

int pod_count;
int whale_count;
species_count(species, first_pod, &pod_count, &whale_count);
printf("%d %s pods containing %d whales in %s\n", pod_count, species, whale_count, filename);
```

Your task in this exercise is to complete this function:

```
void species_count(char species[], struct pod *first_pod, int *n_pods, int *n_whales) {
    // REPLACE THIS COMMENT WITH YOUR CODE
    // THIS FUNCTION SHOULD NOT CALL SCANF OR PRINTF
    // IT SHOULD JUST ASSIGN VALUES to N_PODS AND N_WHALES
    *n_pods = 24; // CHANGE ME
    *n_whales = 42; // CHANGE ME
}
```

Do **not** change any other function.
You are not permitted to use arrays in this exercise.

species_count should assign the number of pods of the given species to `*n_pods` and the total number of whales of the given species to `*n_whales`.

When you have completed the function **species_count** this is how **species_count_list.c** should behave:

```
$ dcc -o species_count_list species_count_list.c
$ ./species_count_list whales.txt "Blue whale"
51 Blue whale pods containing 1171 whales in whales.txt
$ ./species_count_list whales.txt "Indo-Pacific humpbacked dolphin"
43 Indo-Pacific humpbacked dolphin pods containing 897 whales in whale
s.txt
```

When you think your program is working you can use `autotest` to run some simple automated tests:

```
$ 1511 autotest species_count_list
```

When you are finished on this exercise you and your lab partner must both submit your work by running **give**:

```
$ give cs1511 wk08_species_count_list species_count_list.c
```

Note, even though this is a pair exercise, you both must run **give** from your own account before **Sunday 16 September 23:59:59** to obtain the marks for this lab exercise.

Exercise-03: Print A Day's Whale Sightings (pair)

This is a pair exercise to complete with your lab partner.

Here is the starting code for this exercise.
It already contains the functions which you discussed in your tutorial which reads the file of whale sightings into a linked list of structs. You do not need need to change these functions.

The main function in **day_whale_list.c** has this code:

```
struct pod *first_pod = read_sightings_file(argv[1]);
struct date *day = string_to_date(argv[2]);

day_whales(first_pod, day);
```

Your task in this exercise is to complete this function:

```
void day_whales(struct pod *first_pod, struct date *day) {


    // PUT YOUR CODE HERE
}
```

Do **not** change any other function.
day_whales should **print** every whale sighting on the specified day in the order they occur in the linked list of structs.

You should add at least one new function.

You are not permitted to use arrays in this exercise.

When you have completed the function **day_whales** this is how **day_whale_list.c** should behave:

```
$ dcc -o day_whale_list day_whale_list.c
$ ./day_whale_list whales.txt 31/03/18
11 Common dolphin
9 Bryde's whale
11 Dwarf sperm whale
35 Pygmy right whale
3 Common dolphin
20 Long-finned pilot whale
8 Dwarf sperm whale
$ ./day_whale_list whales1.txt 10/05/17
14 Sei whale
34 Indo-Pacific humpbacked dolphin
```

When you think your program is working you can use `autotest` to run some simple automated tests:

```
$ 1511 autotest day_whale_list
```

When you are finished on this exercise you and your lab partner must both submit your work by running **give**:

```
$ give cs1511 wk08_day_whale_list day_whale_list.c
```

Note, even though this is a pair exercise, you both must run **give** from your own account before **Sunday 16 September 23:59:59** to obtain the marks for this lab exercise.

Exercise-04: Print Whale Sightings For a Range of Days (pair)

This is a pair exercise to complete with your lab partner.

Here is the starting code for this exercise.
It already contains the functions which you discussed in your tutorial which reads the file of whale sightings into a linked list of structs. You do not need need to change these functions.

The main function in **between_days_whale_list.c** has this code:

```
struct pod *first_pod = read_sightings_file(argv[1]);
struct date *start_day = string_to_date(argv[2]);
struct date *finish_day = string_to_date(argv[3]);

between_days_whales(first_pod, start_day, finish_day);
```

Your task in this exercise is to complete this function:

```
void between_days_whales(struct pod *first_pod, struct date *start_day, struct date *finish_day) {

    // PUT YOUR CODE HERE
}
```

Do **not** change any other function.
between_days_whales should **print** every whale sighting between the specified two dates, including the dates themselves. The sightings should be printed in the order they occur in the linked list of structs.

You should add at least two new functions.

You are not permitted to use arrays in this exercise.

When you have completed the function **between_days_whales** this is how **between_days_whale_list.c** should behave:

```
$ dcc -o between_days_whale_list between_days_whale_list.c
$ ./between_days_whale_list whales.txt 25/02/18  03/03/18
25/02/18  4 Dwarf sperm whale
03/03/18  8 Pygmy right whale
02/03/18 31 Sei whale
02/03/18 20 Long-finned pilot whale
26/02/18  9 Spinner dolphin
27/02/18 17 Dwarf minke whale
02/03/18 23 Pygmy sperm whale
25/02/18  5 Humpback whale
25/02/18 14 Humpback whale
02/03/18 37 Short-finned pilot whale
01/03/18 35 Long-finned pilot whale
03/03/18 32 Long-finned pilot whale
01/03/18 39 Fin whale
28/02/18 33 Southern right whale
27/02/18  1 Fin whale
01/03/18 24 Humpback whale
25/02/18  4 Long-finned pilot whale
28/02/18  3 Short-finned pilot whale
$ ./between_days_whale_list whales.txt 29/12/17  03/01/18
30/12/17  2 Coastal bottlenose dolphin
29/12/17 24 Dwarf sperm whale
03/01/18 21 Fin whale
29/12/17 34 Pygmy right whale
02/01/18  2 Short-finned pilot whale
29/12/17 12 Striped dolphin
30/12/17  2 Orca
03/01/18 41 Indo-Pacific humpbacked dolphin
03/01/18  1 Dwarf minke whale
30/12/17 29 Striped dolphin
03/01/18  6 Coastal bottlenose dolphin
30/12/17 25 Pygmy right whale
02/01/18  5 Spinner dolphin
02/01/18 33 Short-finned pilot whale
03/01/18 27 Coastal bottlenose dolphin
01/01/18 35 Indo-Pacific humpbacked dolphin
03/01/18 25 Dwarf sperm whale
```

Hint: use the printf format `"%02d/%02d/%02d"` to print dates and `"%2d"` to print how many whales.

When you think your program is working you can use `autotest` to run some simple automated tests:

```
$ 1511 autotest between_days_whale_list
```

When you are finished on this exercise you and your lab partner must both submit your work by running **give**:

```
$ give cs1511 wk08_between_days_whale_list between_days_whale_list.c
```

Note, even though this is a pair exercise, you both must run **give** from your own account before **Sunday 16 September 23:59:59** to obtain the marks for this lab exercise.

Exercise-05: Merge Whale Sightings from the Same Day (individual)[Challenge Exercise]

This is an individual exercise to complete by yourself.

Here is the starting code for this exercise.
It already contains the functions which you discussed in your tutorial which reads the file of whale sightings into a linked list of structs. You do not need need to change these functions.

The main function in **merge_day_whales.c** has this code:

```
struct pod *first_pod = read_sightings_file(argv[1]);

merge_day_whales(first_pod);

write_sightings(stdout, first_pod);
```

Your task in this exercise is to complete this function:

```
void merge_day_whales(struct pod *first_pod) {

    // PUT YOUR CODE HERE
}
```

Do **not** change any other function.
merge_day_whales should merge any subsequent sighting of the same whale species on the same day into the first sighting in the list of that whale species on that day.

The number of whales seen in subsequent sightings should be added to the number of whales seen in the first sighting.

The subsequent sightings (if any) should be deleted from the list and all associated memory freed.

You should add at least one new function.

You are not permitted to use arrays in this exercise.

You are not permitted to use malloc in this exercise, apart from its use in the supplied code.

When you have completed the function **merge_day_whales** this is how **merge_day_whales.c** should behave:

```
$ cat unmerged_whales.txt
18/01/18  9 Pygmy right whale
01/09/17 21 Southern right whale
18/01/18  1 Pygmy right whale
18/01/18  1 Pygmy right whale
16/02/18  4 Striped dolphin
16/02/18  4 Striped dolphin
18/01/18  1 Pygmy right whale
02/07/17  4 Common dolphin
16/02/18  5 Striped dolphin
03/02/18  6 Pygmy right whale
18/01/18  3 Pygmy right whale
$ dcc -o merge_day_whales merge_day_whales.c
$ ./merge_day_whales unmerged_whales.txt merged_whales.txt
$ cat merged_whales.txt
18/01/18 15 Pygmy right whale
01/09/17 21 Southern right whale
16/02/18 13 Striped dolphin
02/07/17  4 Common dolphin
03/02/18  6 Pygmy right whale
```

If **autotest** gives you a **free not called for memory allocated with malloc** error you can reproduce this by compiling and running with `dcc --leak-check`, e.g.:

```
$ dcc --leak-check -o merge_day_whales merge_day_whales.c
$ ./merge_day_whales unmerged_whales.txt merged_whales.txt
Error: free not called for memory allocated with malloc in function rea
d_date in test_merge_day_whales.c at line 161.
```

Make sure when you delete a sighting from the list, you free the memory for the sighting (struct pod) and the whale species name and the sighting date.
Hint: if you don't understand how to delete a struct from a linked list watch COMP1511 tutor Mark Sonnenschein. explain the concepts:

and then Dillon Giacoppo explain the coding:

When you think your program is working you can use `autotest` to run some simple automated tests:

```
$ 1511 autotest merge_day_whales
```

When you are finished working on this exercise you must submit your work by running **give**:

```
$ give cs1511 wk08_merge_day_whales merge_day_whales.c
```

Submission

When you are finished with each exercise make sure you submit your work by running **give**.
You can run **give** multiple times. Only your last submission will be marked.

Don't submit any exercises you haven't attempted.

If you are working at home, you may find it more convenient to upload your work via give's web interface.

Remember you have until **Sunday 16 September 23:59:59** to submit your work.