CST8234 Lab 5

Table of Contents

If you plan on downloading/printing this, note:

- *You will not see any updates/fixes when using a local/physical copy.*
- *A printed html may not display visual artifacts properly.*
- *DO NOT DISTRIBUTE - Copyright of Algonquin College*

0.1 Updates

- Updates will be added here.
  Please don't print/download this document to be able to see them.

1 Due Dates and Deliverables

Due Date:

- See the Brightspace Calendar and under **Activities** -> **Assignments**

Deliverables:

- Upload **`lab05.zip`** file with only the **`*.c`**, **`*.h`** files and **`makefile`** from your VM to Brightspace

Evaluation:

- Weight: 7%

Materials:

2 Topics

- Recursion
- Use of linked lists
- Makefile
- Using structures

3 Procedure

In this assignment you will write a program which puts data into a sorted list

3.1 Plagiarism and Unoriginal Work

As mentioned in the Course Section Information / Weekly Schedule document, any work or algorithm that is not original work created by you will receive a grade of 0.

You may not use outside resources to complete this lab.

Do not share your solution with other students, all students submitting duplicate code will be reported for plagiarism.

3.2 Program Code Requirements

1. Note:
   - All submitted code is expected compile with the **`-ansi -pedantic -Wall`** flags without warnings/errors.
     Heavy grade deductions will take place for not following the C90 standard.
   - It is advised to use **`-g`** and debug your code using the **`gdb`** tool
   - Do not include **`stdbool.h`**, it was introduced in C99
   - Use predefined values from header files (e.g. **`EXIT_SUCCESS`**) and define your own when needed. Do not use "magic numbers".
2. A program named **`lab5`** is used to place information into a list.
   - A linked list (with dynamically allocated memory) must be used as the data structure for this lab.
   - You will use at least two .c files for solving this lab, and header files as needed. The **`lab5.c`** will contain the **`main`** function.
3. Create a **`makefile`** for your program. The makefile will:
   - By default, compile every **`.c`** file with all expected flags/options
   - Have a (phony) target **`run`** to compile and run the code
   - Have a (phony) target **`clean`**, to remove any file created during compilation

*The **`makefile`** will only recompile if there are changes requiring it*

*Tip: This project will include multiple .c files. **It's easy to miss warnings in such project when recompiling!***

*You may want to also add the **`-Werror`** flag to **`gcc`** to treat all warnings as errors.*

3.3 Arguments

4. The program will read data about people using CLI arguments using the following syntax:

```
lab5 [ID NAME AGE]...
```

For example, information on 3 people is given as arguments:

```
$ ./lab5 17000 Yuanyuan 22 14231 Dafni 47 503 Da
```

It is unknown how many people will be provided as input.

Assumptions:

- IDs are always integers
- Names are always ASCII characters
- Ages are always integers

The system is very limited, and has the following restrictions:

- IDs are between **`0`** .. **`99,999`** (inclusive)
- Names are between **`1`** and **`15`** characters long (inclusive)
- The age is always between **`0`** .. **`99`** (inclusive)

5. The program be able to detect the following errors on the arguments:

- IDs, names and ages that won't fit in this system
- Invalid number of arguments

On error, program will report exactly:

- What was the problem
- The argument that caused the problem
- What need to be done to correct the problem
- In case that the number of arguments is invalid, also explain how to use the program.

Afterwards, the program will terminate with exit status **1**.

3.4 Input Processing

6. The program will place all the provided information into a linked list which **automatically sorts the entries** using the **ID** field as the key, from the lowest to the highest value. The entries are sorted by inserting them into the correct "place" in the linked list.

The function which inserts the entries into the linked list **must be a recursive function**.

7. During this stage **ONLY**, using the information in the linked list, the recursive function will detect and report an error if an entry could not be added due to a duplicate ID field.

If it is detected that the same ID is already in the list, the program will terminate and report exactly:

- What was the problem
- The argument that caused the problem
- What need to be done to correct the problem

Afterwards, the program will terminate with exit status **2**.

If any memory could not be allocated, the program will terminate with exit status **3** and an error message will be displayed.

3.5 Output

8. Once all data was added into the self-sorting linked list, the information in each entry will be printed, one entry per line, and the program will terminate.

For example:

```
$ ./lab5 17000 Yuanyuan 22 14231 Dafni 47 503 Da
503 Dan 77
14231 Dafni 47
17000 Yuanyuan 22
$
```

3.6 Memory Allocation/Release

The program must properly free all dynamically allocated memory. Use **`valgrind`** to verify that the program doesn't leak memory, especially when errors are detected.

4 Document your work

Include in the beginning of each C file you submit a comment with:

- Full name
- Student ID
- A paragraph describing the content of the file

Additional comments about the purpose of functions are advised for future reference.

5 Submit your work to Brightspace

> ***IMPORTANT:*** ***`clean`*** *your project and make sure all source files can recompile with the required flags and with no warnings/errors before proceeding!*

In your VM, return to Firefox, where we have the CST8234 Brightspace page open. Under **Activities** -> **Assignments** you will find Lab 5.

`clean` your project using the make file, and zip together the **`.c`**, **`.h`** and **`makefile`** files into **`lab05.zip`**.

Do not include any redundant files or compiled program in the zip file.

> *Extract the zip file into a separate folder and make sure it compiles to avoid losing points!*

Upload **`lab05.zip`** to Brightspace to submit your lab

> - *Make sure that you select the correct file!*
> - *There may be multiple labs posted. Make sure that you submit into the correct Lab!*

You will receive a confirmation email when your lab is submitted.

> ***Always make sure that you receive this email and verify it carefully!***
>
> *This email is your proof that your assignment was submitted on time and lists the files you submitted.*
>
> 1. *Verify the course name*
> 2. *Verify the assignment name*
> 3. *Verify the file name you submitted*
> 4. *Verify the file size*

6 Important Notes on Submissions

- We do not accept any assignment submissions by Email. Always submit your work to Brightspace.

- You can submit multiple times. The last submission will be graded.
- No marks are awarded for submitting under the wrong assignment number or for submitting the wrong file.
