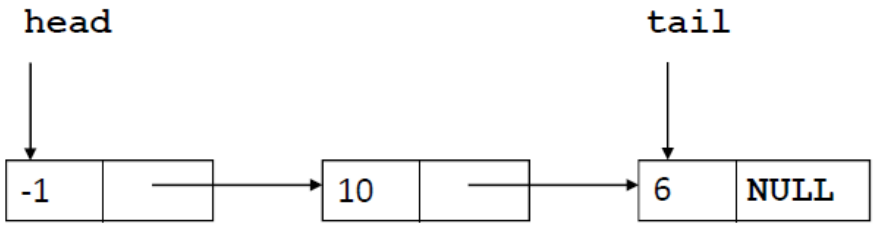
LAB 7 — Pointers to Pointers, File I/O, and UNIX Commands

Problem A – Pointer to Pointers

1. Specification

Refer to the lecture nodes for the description of a singly linked list. In our implementation, the linked list stores non-negative integers. A dummy node with the data value -1 is used to simplify insertions and deletions. See the diagram below.

Write a C program to implement the insertion and deletion operations.



Note: The difference between this lab exercise and problem B of Assignment 1 is that pointer variables `head` and `tail` are no longer global variables. They are now local variables defined inside the `main` function and passed to the insertion and deletion functions.

2. Implementation

- The program to be submitted is named `slist.c`. **Use the given template `slist.c`** and fill in your code. **Submit only file `slist.c`.**
- You are also given a file named `slistMain.c` to test your code. Do not submit file `slistMain.c`.
- The first function to be implemented is `insert()`. See file `slist.c` for its specification. The new element is to be inserted at the end of the list. If a new node cannot be created (e.g., insufficient memory), the function calls function `prtError()` to display an error message and exit the program using `exit(1)`.
- The second function to be implemented is `removeFirst()`. See file `slist.c` for its specification. If the list is empty (i.e., no element other than the dummy node), the function calls function `prtError()` to display an error message and returns -1. Otherwise, it removes the first element (i.e., the node right behind the dummy node) and returns the data (integer) of the removed node.
- You may define your own variables inside functions `insert()` and `removeFirst()`.

- In file `slist.c` you are given three utility functions: `init()`, `prtError()` and `prtList()`. DO NOT modify these functions.
- Do not modify the function and structure definitions in file `slist.c`.
- Assume that all inputs are valid. No error checking is required on inputs.
- To compile both files `slist.c` and `slistMain.c`, use the following command:

```
cc slist.c slistMain.c
```

3. Sample Inputs/Outputs

See file `slist_out.txt` for the output from running programs `slist.c` and `slistMain.c`.

Problem B – File I/O

1. Specification

Write a C program to input student records from a file. Each entry of the input file has the following format:

[First name] [Last name] [Assignment 1 mark] [Assignment 2 mark]

Copy the input data to an output file, and add one more field that records the average of the two assignment marks. So each entry of the output file has the following format:

[First name] [Last name] [Assignment 1 mark] [Assignment 2 mark] [Average mark]

2. Implementation

- The program to be submitted is named `marks.c`. **Use the given template `marks.c`** and fill in your code. **Submit only file `marks.c`.**
- To compile the program, use the following command: **`cc marks.c –o marks`**
- The file names are input as command line arguments. File names are less than 30 characters long.
- Sometimes users may forget the command syntax and they may type only the command "`marks`", or enter only one file name. In that case, display the following reminder message on the standard output:

```
Usage: marks [input_file] [output_file]
```

- Assignment marks are integers. Average marks are of type `float`. Output the average marks with one decimal digit.

- If a file cannot be opened for read or write, display an error message and exit the program using `exit(1)`.
- Use `fscanf()` to read from the input file and `fprintf()` to write to the output file.
- You may define your own variables inside function `main()`, and implement your own function(s) in file `marks.c`.
- Assume that students' names and marks to be input are valid and no error checking is required.

3. Sample Inputs/Outputs

See file `marks_exe.txt` for a snapshot of program execution, and contents of the sample input and output files.

Common Notes

- Complete the header in files `slist.c` and `marks.c` with your student and contact information.

Problem C – UNIX Commands

See the next page.

Lab 7 - UNIX Commands

1. The **echo** command takes a line that you type and repeats it back on the screen. Redirect your input line to a file named **myFile**.
2. Use the **echo** command to input 5 more lines and redirect them to file **myFile**, which will contain a total of 6 lines of text.
3. Using the commands **who**, **who am i**, and **date**, append to file **myFile** the outputs of the above commands.
4. Suppose **myFile** is in your working directory. Specify the command(s) you would use to do the following:
    a. Give everyone permission to read **myFile**; do not change any other privileges.
    b. Permit the owner (you) and group members to read and write the file; remove all privileges from everyone else.
    c. Remove writing privileges from everyone but the owner.
    d. Give the owner and group members permission to execute **myFile**, while giving the owner sole permission to read and write the file.
5. Suppose you have a directory named **myStuff** in your working directory. Specify the command(s) you would use to do the following:
    a. Give everyone permission to list files in **myStuff**; do not change any other privileges.
    b. Permit the owner and group members to list, remove, or add files; remove all privileges from everyone else.
    c. Remove writing privileges from everyone but the owner.
    d. Give the owner and group members permission to execute **myStuff**, while giving the owner sole permission to read and write the directory.
6. Create a set of text files and name them *backgammon, backpacking, baseball, boxing, biking, chess, fencing, blackjack, groupA, groupB, groupX, groupY*.
    a. How would you use **cat** to show the contents of the files ending in *ing*?
    b. How would you list any files containg *x* or *X* in the file names?
    c. How would you show the contents of files with names containing *o*?
    d. How would you show the contents of the files *backgammon, backpacking* and *blackjack* using just one command?
    e. How would you copy the contents of all 3 files *backgammon, backpacking* and *blackjack* to a file named *all3* using just one command?
7. Given the file *phone_book.txt*, specify the command(s) you would use to do the following:
    a. Display entries of people whose names contain pattern *alex*, case-insensitive. (The output should be the entries of *alex johnson* and *Alexander Smith*.)
    b. Count the number of people whose area code is 905. Display just the count (not the entries).
    c. Display the entries of the babysitters.
    d. Count the number of friends whose names are in *phone_book.txt*. Display just the count (not the entries).

Save your commands in a file named *lab7_unix.txt* containing the following header and submit file *lab7_unix.txt*.

```
# EECS 2031 - Lab 7
# Filename: lab7_unix.txt
# Author: Last name, first name
# Email: Your email address
# Login ID: Your login ID
```