CSE 2451 Lab 2

Objectives

- linked-list
- dynamic memory allocation
- File I/O

Introduction

In this lab, you will implement linked list to store data you read from a file stream. The list node struct is given by the following code snippet:

```c
struct list_node {
    char name[40]; // store the student osu id
    double grade; // store a floating-point number
    struct list_node *next; // the next pointer in linked list
};
typedef struct list_node Node;
```

The following functions are provided in the src code template on carmen for this lab:

```c
// create a list node with malloc
Node * create_node(char * name, double grade);
// pack input data (name, grade) as a new node to front of the linked list
Node * push_data(char * name, double grade, Node * list);
// print a given linked list
void print_list(Node * list);
```

Note: any memory allocation failure will result in abortion of the program, please see the source code template for more details.

Question 1 delete linked list (2 pts)

Implement a function to free allocated memory for a linked list. You need to traverse through the linked list and free allocated memory block for each node. Please follow the function prototype given below:

```c
// list is a pointer to the front node of the linked list
void delete_list(Node * list);
```

Question 2 grade statistics (4 pts)

For a given linked list constructed from the aforementioned Node, traverse through the list and return the associated sample mean and sample standard deviation.

$$\hat{\mu} = \frac{1}{N}\sum_{i=0}^{N-1} g_i$$

$$\hat{\sigma} = (\frac{1}{N}\sum_{i=0}^{N-1}(g_i - \hat{\mu})^2)^{\frac{1}{2}}$$

Note: you need to use **double sqrt(double);** from **math.h** library to calculate the square root. For historical reasons, gcc doesn't link the math library by default (unlike libararies for stdlib.h, or stdio.h). Therefore, you need to use the link option **-lm** to instruct gcc explicitly to make sqrt() work. Please follow the function prototype given below:

```
/*
return number of nodes in the linked list
list: a pointer to the front node of the linked list
mean: a pointer to a double variable in the caller's scope
std: a pointer to a double variable in the caller's scope
here we apply the "passing by reference" with pointer trick to return 3 values
from one function. mean and std are used to return grade statistics to
variables in the caller's scope
*/
int grade_stats(Node * list, double * mean, double * std);
```

Question 3 file to list (4 pts)

For a given text file, read the file with file I/O functions (fopen, fscanf, fclose) while saving the associated data in the text file to the linked list used in this lab (I've already implemented push_data() for this purpose in the source code template). You may assume the input file have multiple lines, all lines share the same format: $< name > < grade >$ where $< name >$ includes alphabetic letters and $< grade >$ is composed of numeric characters and the decimal point. For example:

```
Tom 93.5
Adam 79.8
Nick 99.7
```

The actual text file will be provided on carmen. Please download the file and use it for testing your code. Please follow the function prototype given below:

```
// file_path is the file path of the text file you want to read
// return a pointer to the front of the linked list
// you may use push_data() provided in the source code template
// to implement this function
Node * file_to_list(char * file_path);
```

Hint: (1) use push_data() to an empty list will create a list with one node.

Bonus: minimum grade (2 pts)

Implement a function to traverse through the linked list and return the minimum grade stored in the list. Please follow the function prototype given below:

```c
// list is a pointer to the front node of the linked list
// return the minimum grade stored in linked list
double min_grade(Node * list);
```

Bonus: maximum grade (2 pts)

Implement a function to traverse through the linked list and return the maximum grade stored in the list. Please follow the function prototype given below:

```c
// list is a pointer to the front node of the linked list
// return the maximum grade stored in linked list
double max_grade(Node * list);
```

Bonus: the apply function (2 pts)

Implement an apply function that will take a pointer to Node and a function pointer (pointing to either min_grade() or max_grade()) as input arguments, and apply the associated function pointed by the function pointer to the input linked list. The associated function prototype should have a pointer to Node as its first input argument and a function pointer as the second input argument, and return a double typed value.

```c
// list is a pointer to the front node of the linked list
// you need to figure out what's the proper way to declare
// an associated function pointer that points to
// functions of with "double function_name(Node *)" prototype;
// return a double typed value
double apply(Node * list, ?);
```

Submission Requirement

1. download source file template (src_template.c) from carmen and rename the file to "lab2.c"
2. download text.txt file in carmen and put it in the same directory with your source file
3. fill in your implementation of the associated functions. **Do NOT change the test cases**, just uncomment them once you finish the associated functions
4. compile your source code with the following command (-lm will link library for math.h):

```
gcc -Wall -g lab2.c -o prog -std=c99 -lm
```

5. check if your program have memory errors with the following command, you don't need to submit the valgrind log, but memory errors will lead to penalty on your grade

```
valgrind --leak-check=full ./prog
```

6. if you only completed part of the lab assignment, just include those completed functions in your final source file and comment out the incomplete functions so that your source file can still compile.
7. upload one file: your source code "lab2.c" file.