

EEL 4851
Engineering Data Structure

Summer 2005

Home Work Schedule

Schedule of Assignments:

	Week		
<u>HW#</u>	<u>Due</u>	<u>Points</u>	<u>Title</u>
1	Due	2%	Stacks, queues and the list using linked list
2	7/12/05	4%	Identifier index (AVL search trees)
3	8/02/05	4%	Direction finder

WARNING:

I WILL NOT ACCEPT ANY HOMEWORK AFTER THE LAST DAY OF CLASS.

All homework must include:

- A Cover page with the following information
 - Course number (EEL 4851)
 - Your name
 - Home work number
- Short description of the work done.
- Source code with comments.
- Program output.

Notes:

- Comments on the source code are to explain what the code does. This will help the grader (TA) as well as your self understand how your code works and what each function does. Each function must have comments indicating exactly what that function does.
- Homework assignments are due at the BEGINING of the first class meeting for the week it is due. Late assignments will be accepted but will be penalized at 10 % per week late. I will not accept any homework after the last day of class.
- Homework assignments are to be turned in to me (course instructor, Dr. Gonzalez) or put into my mailbox or under my office door (enr 211).
- You are required to keep a copy of all material incase it gets lost. You must also keep all graded material returned to you to prove it has been turned in on time.

Homework Assignment #1

EEL 4851 Engineering Data Structure

Topic: Stacks, queues and the list data structure using linked list.

Stacks, queues and the list using linked list

Write a list class to implement a doubly linked list. The list should accommodate integers and pointers. You may add other types as you need them.

Include member functions for the following:

```
void      insert_front(int data );
void      insert_front(void *data );
void      insert_rear(int data );
void      insert_rear(void *data );
int       remove_front_i( );
void *    remove_front_p( );
int       remove_rear_i( );
void *    remove_rear_p( );
int       empty( );
```

Use dynamic memory. Use the "new" instruction to allocate memory when inserting and the delete instruction to return the memory when removing. Use the union type found in C for the data field of the node so that it can accommodate both data types.

Make the list into a class where the actual list is private and the member functions are public. Prove your program correctness by applying several test cases. Its up to you to decide what test cases will provide sufficient confidence your program works correctly. The output should have several test cases. Do a good job as you will use this class in the remainder of your programming assignments.

Homework Assignment #2

EEL 4851 Engineering Data Structure

Topic: AVL trees.

Identifier index

Write a program that reads a C++ source code file and outputs a list of all identifiers (that is, all variable names that are not keywords and that are not found in comments or string constants) in alphabetical order. Each identifier should output with a list of line numbers on which it occurs. See HW 4.50 on page 176. Use an AVL binary search tree to store the identifiers seen. The program must work for arbitrary large files sizes. Each node should have a linked list to store all of the line numbers. Use the list you created in assignment 1. Make the tree a class with the necessary functions.

Homework Assignment #3

EEL 4851 Engineering Data Structure

Topic: Graph and Sorting algorithms.

Finding direction.

You are to write a program that finds the driving direction of a specified initial and end location much like MapQuest. You are to read the map information from a text file and create a graph. Each intersection and point of interest in the map will be given a unique name and be a vertex in the graph. Each road connecting 2 intersections is an edge in the graph. The edge names do not need to be unique. Use a directional graph since streets may be one-ways. The text file will look like:

Alafaya&GeminiN	Gemini	Gemini&GreekParkCt	East	.3	35
Gemini&GreekParkCt	Gemini	Alafaya&GeminiN	West	.3	35
Gemini&GreekParkCt	Gemini	Gemini&KnightCtE	East	.5	35
Gemini&KnightCtE	Gemini	Gemini&GreekParkCt	West	.5	35
Gemini&KnightCtE	KnightCt	Arena	North	.1	20
Arena	KnightCt	Gemini&KnightCtW	South	.1	20

Steps:

1. Write a function to perform a quick sort on a list of alphanumeric data.
2. Read the text file and add each intersection name to the list. Only read the first column since all intersection names will be in there.
3. Sort the list using the Quick Sort or Merge Sort algorithm
4. Copy the names from the sorted list into the graph. Omit duplicates.
5. Create the graph by reading the file and adding an edge for each street. Use a binary search to find the proper vertex.
6. In the main program ask the user to input a start and end intersection.
7. Find the shortest and quickest path from the start to the end intersection. The output should indicate the name of the street and the distance.

The output should look like:

From Alafaya&GeminiN

Take Gemini East to Gemini&GreekParkCt	.3
Take Gemini East to Gemini&KnightCtE	.5
Take KnightCt North to Arena	.1

The shortest path is the path with the least mileage while the fastest path is the path with the least time. The time is the distance multiplied by the MPH.

You should have at least 20 intersections and 30 roads. You may share map files with other students. In fact if you thing you have a good map file allow me to have a copy for future students.

Hint the node structure for the graph should include the street name, direction like East, dist, and MPH. The Graph array should include the intersection name and the pointer to the linked list.

