University of Missouri – Computer Science 2050

DUE FRIDAY, NOVEMBER 18 AT 11:59PM

Homework 3

Introduction

In this homework you will be given an input file with some vehicles and some information about those vehicles. You will parse through the input file and put the information in Node structures. You will add these nodes to a binary search tree based on price of the vehicles. After the tree has been built, you will give the user the ability to search by price (see sample output). You also must use a make file to compile this homework. The text files needed will be provided for you. Each variable of a Node structure has a prefix that lets you know what it is they are respectively ‘make’, ‘model’, ‘price’, ‘year’, ‘color’ ‘type’, ‘mileage’. The reason they need prefixes is because the information in the input file for each car is in no specific order. Each string value (make, color, type) has a corresponding input file that contains valid entries. If the string in the input file is not found in the corresponding file of valid types, the car is not valid and should not be added to the tree. For example, if the input file contains ‘make|yellow’, since yellow is not a value in the makes.txt file, the car is not valid and should not be added to the tree. **There will always be an entry for every value in the input file but the entry may not be valid (**color|turquise|type|suv|model|camaro|**make|fake**|mileage|54507|price|26784|year|1979|). On a side note, the text files you use may be hardcoded into the get\_cars function. **The cars should be printed in order of price and the number of cars found should be printed after the cars have been ouput. You MAY NOT change the prototypes or use global variables. You MAY add functions that are not included if you so feel.**

File Breakdown

You are free to break up the program into as many files as you would like. The only requirement is that you have at least 2 .c files and at least 1 .h file. A good way to do it would be to put main in one .c file and all the function definitions in a separate .c file. The .h file will contain all the libraries you need and the function and structure prototypes. The .c files you create should only #include the .h file you have created

Function Prototypes

#define MAXLINELENGTH 200

#define MAXCARPRICE 1000000

typedef struct node {

char make[25];

char model[25];

int price;

int year;

char color[25];

char type[25];

int mileage;

struct node\* left;

struct node\* right;

} Node;

You will need a function to get the valid values for each of the string characteristics. It is up to you on how you want to do this.

// This function takes the file that has all the cars stored and scans in

// their attributes into a Node structure. The function then calls the add\_to\_tree

// function to add the cars to the tree.

// Parameters

//

// file - This is the file that the vehicle information is originally

// stored in

//

// This function returns the head of the tree that the vehicles are stored in

//

Node\* get\_cars(char\* filename);

// This function takes a node in and adds it to the binary

// search tree

//

// Parameters

//

// head - The head of the binary search tree

//

// new - The new node to be added to the binary search tree

//

// This function returns the head of the binary search tree

Node\* add\_to\_tree(Node\* head, Node\* new);

// This function will search the tree given the parameters

// and print out the vehicles that fall under the parameters

//

// Parameters

//

//root - The root of the binary search tree

//

// year - The year the user wants the vehicles to be newer than

// if this is not specified it should default to 0

//

// price - the price the user wants the vehicle to cost less than.

// if not specified it should default to the define statement given above

// count – the number of cars that have been output

void search\_tree(Node\* root, int year, int price, int\* count);

Other helpful functions:

* fgets
* strtok(need string.h library)

Sample Output

NOTE THAT THIS OUTPUT IS A PIECE OF WHAT IT SHOULD BE. (There should be more cars returned than what is shown).

[ajwvz9@tc-login HW3]$ make

compile HW3.c functions.c -I.

[ajwvz9@tc-login HW3]$ ./a.out

Correct input: ./a.out <input file>

[ajwvz9@tc-login HW3]$ ./a.out input.txt

To search for a vehicle newer than a specific year enter '1'

To search for a vehicle that costs less than a certain price enter '2'

To search for a vehilce with both of the above parameters enter '3'

To exit enter '-1'

1

Enter the year you would like the vehicles to be newer than: 2015

2016 dodge escape that costs $867

2016 jeep 350z that costs $1631

2016 jeep 350z that costs $4094

2016 subaru forester that costs $5237

2016 bmw malibu that costs $6169

2016 acura escape that costs $8652

There were 69 cars found

To search for a vehicle newer than a specific year enter '1'

To search for a vehicle that costs less than a certain price enter '2'

To search for a vehilce with both of the above parameters enter '3'

To exit enter '-1'

2

Enter the price you would like the vehicles to be less than: 2500

1952 mercury mal that costs $0

1959 toyota rubicon that costs $1

1963 pontiac sierra that costs $2

2 honda rubicon that costs $3

2 toyota navigator that costs $4

1979 kia camaro that costs $6

1961 toyota slk that costs $8

There were 189 cars found

To search for a vehicle newer than a specific year enter '1'

To search for a vehicle that costs less than a certain price enter '2'

To search for a vehilce with both of the above parameters enter '3'

To exit enter '-1'

3

Enter the year you would like the vehicles to be newer than: 2015

Enter the price you would like the vehicles to be less than: 2000

2016 dodge escape that costs $867

2016 chevrolet sable that costs $1251

2016 jeep 350z that costs $1631

There were 3 cars found

To search for a vehicle newer than a specific year enter '1'

To search for a vehicle that costs less than a certain price enter '2'

To search for a vehilce with both of the above parameters enter '3'

To exit enter '-1'

-1

[ajwvz9@tc-login HW3]$

Grading Criteria

main . . . . . . . . . 8

get\_cars . . . . . . 12

add\_to\_tree . . . . . 15

search\_tree . . . . . 15

Makefile. . . . . . . 10

------------------------

total . . . . . . . . 60

Submission instructions

zip <nameofzipfile> <filestozip>

submit CS2050 HW3 <nameofzipfile>

Other Notes

* PROGRAMS THAT DO NOT COMPILE WILL BE AN AUTOMATIC 0
* PROGRAMS WITH MEMORY LEAKS WILL BE AUTOMATIC 25% DEDUCTION
* PROGRAMS WITH SEGMENTATION FAULTS WILL BE AN AUTOMATIC 50% DEDUCTION