Seth Cram

CS-121

Program 2: linked list (Program Design)

02/12/2020

In the animal\_linked\_list.h program, include a node class that contains the public function prototypes setnext(), setdata(), print(), append(), clear(), and count(). Also include private node pointer next and private string animal\_name. All of the functions are void because none of them return a value as they all deal with the linked list that you create. Define these functions below as functions of the node class. Since setnext() is used for setting the next address value of a node. It’ll take a node pointer, and sets next to that node pointer. To set the data values in each node, we need the setdata() function that takes a string as an argument since the animal names are strings. The print() function takes no argument as it just traverses the list recursively when called by the head pointer, outputting each node’s animal name as it walks along. Next is the append() function that takes two node pointers as arguments. It takes the head pointer and the node that’s going to be added to the end of the list. This function is used to add each animal to the list, besides the first node that’s supposed to be created in main. It also walks recursively down the list to the end, when next is NULL, and connects the new node. Function clear() accepts a node pointer and string as an argument. The node pointer should be head and the string is the name of an animal. This particular function searches through the linked list, comparing the taken name to each node’s animal name to find the correct node to delete. As it walks along the list it details the previous node as the ‘previous’ node pointer, and the node you’re currently on as the ‘current’ node pointer. When it finds the right node it utilizes the previous, current, and head pointer nodes to relink the list, cutting out the connection to the found node and going around it, before deleting the found node. It gets rid of the found node by setting it’s next to NULL and deleting the node. If clear() doesn’t find a node that matches the given name, it outputs an error message. Lastly, the count() function receives a node pointer that will be head, and a string that will be the name of an animal. It acts like the clear() function in the respect that it walks down the list non-recursively. When it finds a node that matches the name passed to the function, it adds a value to the counter defined within the function that’s outputted at the end of the function’s traversal.

Within the main() of test\_animal\_linked\_list.cpp, every string passed into the function arguments is an element of a string array that is read in from the animal.txt file. Therefore, for loops are used to insert the name strings into the clear() and append() functions. A string array reads in the operations that precede each animal name within the txt file with a file loop. A for loop with an if statement embedded takes care of whether to append() or clear() with the help of the string operation array.