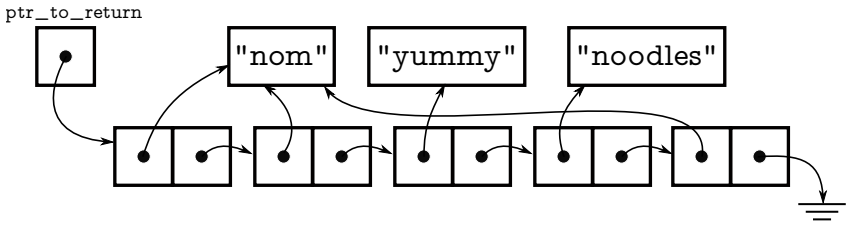
CSCE121: Introduction to Program Design and Concepts

Practice Questions for Final

April 26, 2018

Question 1: Split a sentence into words (50)

You are given a piece of text as a '`\0`' terminated `char[]` and you need to split it into words that are then stored in a linked list. The idea is to save memory in representing strings with a lot repetitions by reusing repeated words. The following figure shows the intended output given the string "nom nom yummy noodles nom"



Use the following type to store each word:

```
struct wrd_t {
    char *str;   // A pointer to the string data
    wrd_t *next; // Next word, or NULL.
};
```

The text for each word must be copied into an array of `char`s that is dynamically allocated to be just the right size (including a '`\0`'). The field `str` is then set to point to it. When the input is a word that has already been encountered, you should reuse the previously allocated memory.

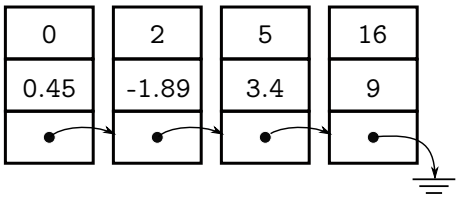
1. Write a function called `split` that returns a pointer the head of the linked list structure storing the words. The declaration is `wrd_t* split(char s[])`
2. Write a function that, given a pointer to the head of the linked list, prints the sentence. Name the function `print_list`.
3. Write a function `clean_up` that will clean up the linked list structure returned from `split`. If you execute `clean_up(split("any sentence here any any"))` no memory should have been leaked.

Question 2: Linked list polynomials (40)

In this question you need to write a class `Poly` to manipulate large polynomials. Suppose the following polynomial is given:

$$0.45 - 1.89x^2 + 3.4x^5 + 9x^{16}$$

We'd manipulate this polynomial in the code below. Internal to the class, it would be stored in a linked list representation like that shown to the right.




```
int main() {
    Poly p;
    p.add_term(0.45, 0);
    p.add_term(1.89, 2);
    p.add_term(3.4, 5);
    p.add_term(9, 16);
    cout << p.eval(0.0) << endl; // Should output 0.45
    cout << p.eval(1.0) << endl; // Should output 10.96
    return 0;
}
```

1. Write a class `Poly` that has both `add_term` and `eval` methods.
2. Ensure that your class has destructor to cleans up allocated memory.
3. Add a method that determines whether two polynomials have the same degree. In the example above, `p.eq_deg(q)` should return `true` if and only if `q` is of degree 16. (That is, the term with non-zero coefficient possessing greatest degree in `q` is x^{16}).

Question 3: Recursive maximum (25)

Here is a declaration of a class that allows one to define a tree:

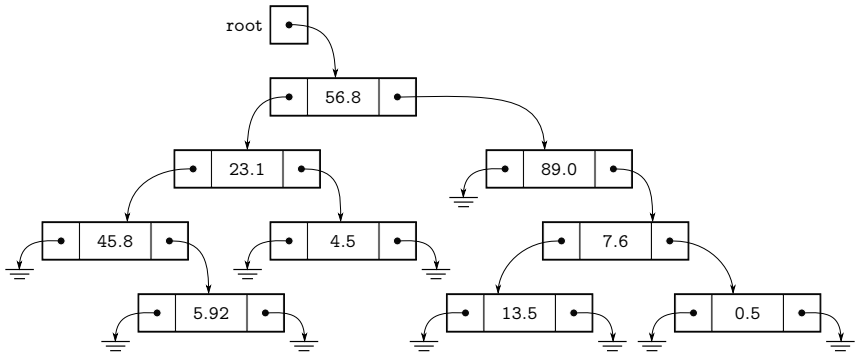
```
class TreeNode {
    public:
        TreeNode(TreeNode *l_child, TreeNode *r_child, double v);
        double get_value();
        TreeNode *get_left();
        TreeNode *get_right();

    protected:
        double val;      // Value stored at this node
        TreeNode *left;  // Pointer to my left child, NULL if none
        TreeNode *right; // Pointer to my right child, NULL if none
};
```

Write a function which, given a pointer to the root (i.e., the top) of a tree, returns the largest element in the tree. Assume that all the values are non-negative numbers. Here is the appropriate declaration:

```
double tree_max(TreeNode *tree)
```

For example, given that `root` points to the following structure, `tree_max(root)` should return `89.0`



Since all the values are non-negative, have `tree_max(...)` return `0` if the pointer is `NULL`.

Important hint: The title of this question is *recursive* maximum.

Question 4: Is there a loop in my linked list? (30)

The following defines a node in a singly linked list.

```
struct item_t {
    string str;   // Some data we're storing
    item_t *next; // Linked item in list
};
```

A friend is using this definition but thinks there's a bug in their code because one of their functions seems to run forever. You suspect that one of the `next` pointers is referring back to something earlier. Write a function that, given a pointer to the head of a list of `item_t`s, determines whether the list has a loop in it.

Question 5: Vehicles and registration taxes (40)

Imagine that you would like a class to refer to various vehicles for computing registration taxes. Classes are used to describe vehicles which bear a licence plate, and should have a method `setReg()` that sets the licence string. A function `getReg()` should return the associated licence string. Additionally, the function `getFee()` should return a `float` that is the annual renewal fee in dollars. The following code should be valid:

```
int main(int argc, char **argv) {
    Trailer *oneHorseSlant = new Trailer();
    Car *lincolnCont = new Car("TX567");
    ElectricCar *tesla = new ElectricCar("TX945");
    oneHorseSlant->setReg("TX642");

    printTaxDetails(oneHorseSlant);
    printTaxDetails(lincolnCont);
    printTaxDetails(tesla);

    delete oneHorseSlant;
    delete lincolnCont;
    delete tesla;
}
```

This would output the following when run:

```
TX642 Tax for trailers is $50
TX567 Tax for cars is $200
TX945 Tax for electric cars is $100
```

The particular fee is associated with each *type* of vehicle, and the fee should be read only.

1. Define a class that is appropriate for vehicles.
2. Define `Trailer`, `Car`, and `ElectricCar` classes, with `ElectricCar` being a subclass of `Car`. Note from the example the way these are created.
3. Create the function `printTaxDetails` to print the license string followed by the renewal fee of any vehicle passed to it. Look at the example output above.

END OF THE PRACTICE QUESTIONS