Computer Science 2530
March 26, 2020

Happy Thursday, March 26.

Answers to the practice questions for exam 3 are now posted on the course web page.

Introduction

Today's lectures (**32A** to **32C**) cover more examples of writing function definitions for linked lists by

1. Writing equations using conceptual list notation, and
2. Converting the equations to a C++ function definition.

The most important takeaways are the following.

1. You usually want to have a separate equation to handle an empty list. For example, **32A** defines function **squares**, and includes equation
$$squares([]) = [].$$
2. You also need at least one case to handle a nonempty list.
   (a) Every nonempty list has a head and a tail. You usually want to get the head and tail of the parameter list, and use them in your equation(s).
   (b) If the result of your function is a list, you can break finding the result into (1) finding the head of the result, and (2) finding the tail of the result.

To understand how to handle a nonempty list, **start with an example**. For **squares(*x*)**, we can use example $x = [3, 5, 6, 7]$.

$$\text{squares}([3, 5, 6, 7]) = [9, 25, 36, 49].$$

Where did the head of the answer, 9, come from? It is the square of the head x. Let's write that down.

$$\text{head}(\text{squares}(x)) = (\text{head}(x))^2.$$

Where did the tail of the answer, [25, 36, 49], come from? It is squares([5, 6, 7]). That is, it is squares(tail(x)). Lets write that down.

$$\text{tail}(\text{squares}(x)) = \text{squares}(\text{tail}(x)).$$

Keep in mind that notation $h\!:\!t$ means "the list whose head is h and whose tail is t." If we know the head and tail of a list, we just put a colon between them to get the whole list. So, when x is not an empty list,

$$\begin{aligned}\text{squares}(x) &= \text{head}(\text{squares}(x)) \;\mathbf{:}\; \text{tail}(\text{squares}(x))\\ &= \text{head}(x)^2 \;\mathbf{:}\; \text{squares}(\text{tail}(x)).\end{aligned}$$

That leads to the following C++ definition of **squares**.

```
ListCell* squares(const ListCell* L)
{
  if(L == emptyList)
  {
    return emptyList;
  }
  else
  {
    int h = head(L);
    return cons(h*h, squares(tail(L)));
  }
}
```

Reading and exercises

Read **32A**. Do the exercises at the bottom of the page.

Read **32B** and **32C**. Do exercises 1-5 at the bottom of page **32C**.