Computer Science 2530
March 25, 2020

Happy Wednesday, March 25.

With this lecture, we begin to look at defining functions that work on linked lists. This page covers pages **31A** to **31D** in the lecture notes.

The key takeaway is you can write a C++ definition of function f by:

1. first thinking about conceptual lists and concentrating on facts about f;
2. then converting the facts to a C++ definition of f.

Using that two step process works much better than trying to work in C++ from the beginning.

1. Equations about lists (31A)

In mathematics, we write $A = B$ to mean that expressions A and B have the same value. $A = B$ is called an *equation*.

Here are some equations that you are familiar with from algebra. Each is true for all of the values of the variables that they mention.

$$\begin{aligned} x + y &= y + x \\ xy &= yx \\ x(y + z) &= xy + xz \\ (x + 1) + 1 &= x + 2 \end{aligned}$$

Now that we have operator **:** and functions head and tail, we can write equations about lists.

$$\text{head}(h : t) = h \tag{1}$$
$$\text{tail}(h : t) = t \tag{2}$$

Let's check equation (1) for the case where $h = 3$ and $t = [5, 7]$.

$$\begin{aligned} \text{head}(3 : [5, 7]) &= \text{head}([3, 5, 7]) \\ &= 3 \end{aligned}$$

Some equations only hold in particular circumstances. We have seen Euclid's equations for the greatest common divisor (gcd) function.

$$\begin{array}{lcll} \gcd(x,0) & = & x & (\text{provided } x \neq 0) \\ \gcd(x,y) & = & \gcd(y, x \bmod y) & (\text{provided } y \neq 0) \end{array}$$

Each equation has a *proviso* saying when it is true.

Suppose we want to define length(x) to be the length of list x. For example, length([2, 4, 6]) = 3 and length([7, 6, 5, 4, 3]) = 5. Two facts are clear.

1. The length of an empty list is 0.
2. The length of a list x is one bigger than the length of the tail of x. For example, length([2, 4, 6, 8]) = 1 + length(4, 6, 8]).

Those facts are easy to express as equations.

$$\begin{array}{lcll} \text{length}([\,]) & = & 0 & \\ \text{length}(x) & = & \text{length}(\text{tail}(x)) & (\text{provided } x \neq [\,]) \end{array} \tag{3}$$

Exercises

Read page **31A**. Do all of the exercises at the bottom of page **31A**.

2. Equations as algorithms (31B)

Equations (3) can be used to compute the length of a list by performing substitutions. Here is a sequence of substitutions showing that length([2, 4, 6]) = 3.

$$\begin{array}{rcl} \text{length}([2,4,6]) & = & 1 + \text{length}(\text{tail}([2,4,6]) \\ & = & 1 + \text{length}([4,6]) \\ & = & 1 + (1 + \text{length}(\text{tail}([4,6]))) \\ & = & 1 + (1 + \text{length}([6])) \\ & = & 1 + (1 + (1 + \text{length}(\text{tail}([6])))) \\ & = & 1 + (1 + (1 + \text{length}([\,]))) \\ & = & 1 + (1 + (1 + 0)) \\ & = & 3 \end{array}$$

That algorithm can be converted into a C++ algorithm. Since there are two equations for length, there are two cases, and that means we need one if-statement to select a case.

```
int length(ListCell* L)
{
  if(isEmpty(L))
  {
    return 0;
  }
  else
  {
    return 1 + length(tail(L))
  }
}
```

We have defined type List to be the same as ListCell*, so the definition of length can be written as follows.

```
int length(List L)
{
  if(isEmpty(L))
  {
    return 0;
  }
  else
  {
    return 1 + length(tail(L))
  }
}
```

Exercise

Read page **31B**. Do the exercise at the bottom of the page.

3. More examples (31C and 31D)

Read pages **31C** and **31D**. They derive definitions of the following functions.

1. member(x, L) returns true if integer x occurs in list L. For example, member(5, [2, 4, 5, 6]) is true, but member(5, [2, 4, 6]) is false.

2. removeAll(x, L) returns the result of removing all occurrences of integer x from L. For example

   (a) removeAll(3, [2, 3, 4, 5]) = [2, 4, 5]

   (b) removeAll(2, [2, 3, 2, 6]) = [3, 6]

   (c) removeAll(4, [4, 4, 4]) = [ ]

   (d) removeAll(5, [2, 3, 4]) = [2, 3, 4]

Page **31D** defines the following equations for removeAll.

$$\begin{array}{lcll} \text{removeAll}(x, [\,]) & = & [\,] & \\ \text{removeAll}(x, L) & = & \text{removeAll}(x, \text{tail}(L)) & (\text{provided head}(L) = x) \\ \text{removeAll}(x, L) & = & \text{head}(L) : \text{removeAll}(\text{tail}(L)) & (\text{provided head}(L) \neq x) \end{array}$$

Here is an evaluation by substitution of removeAll(2, [2, 30, 2, 60]) using those equations.

$$\begin{array}{lcl} \text{removeAll}(2, [2, 30, 2, 60]) & = & \text{removeAll}(2, [30, 2, 60]) \\ & = & 30 : \text{removeAll}(2, [2, 60]) \\ & = & 30 : \text{removeAll}(2, [60]) \\ & = & 30 : (60 : \text{removeAll}(2, [\,,\,])) \\ & = & 30 : (60 : [\,,\,]) \\ & = & 30 : [60] \\ & = & [30, 60] \end{array}$$

Exercises

Solve the following exercise and email me your solution. For this exercise, just write your answer in the body of the email. Send it as soon as you are done with it. I will let you know whether it is correct.

Doing this is essential for you to do well on exam 4. If you don't get feedback before the exam, you will not know whether you understand this material.

Suppose nth(n, L) is supposed to return the n-th value in list L, numbering from 1. For example,

$$\text{nth}(1, [10, 20, 30, 40]) \;=\; 10$$

$$\begin{aligned} \text{nth}(2, [10, 20, 30, 40, 50]) &= 20 \\ \text{nth}(3, [50, 100, 6, 18]) &= 6 \\ \text{nth}(3, [80, 60, 40, 22, 5]) &= 40 \end{aligned}$$

Assume that n is a positive integer and list L has at least n values in it, and do not worry about what happens if that is not true.

(a) Write equations, with provisos, that are true for nth(n, L). It is not enough to give a few examples, Your equations must be general enough so that they can be used to compute nth(n, L) for any positive integer n and list L that has at length at least n.

If what you write does not have the form of equations, it cannot be right.

(b) Using your equations, show an evaluation by substitution of nth(3, [10, 20, 30, 40, 50]).

(c) Convert your equations into a C++ definition of **nth**. Follow the equations directly. Do not throw them away and start over.