Recursion template

Wednesday, March 8, 2023 12:27 PM

Notice that the body of every recursive function on Lists has:

- A cases expression that checks the shape of the list
- An simple, non-recursive computation for the empty case
- A recursive call to rst
- A computation on fst
- A way to combine the fst and rst computations

```
cases(List) num-list:
    | empty => 0
    | link(fst, rst) => fst + sum(rst)
  end
```

```
cases(List) num-lst:
  | empty => empty
  | link(fst, rst) => link(fst * 2, double(rst))
end
```

```
cases(List) col-lst:
  | empty => rectangle(0, 0, "solid", "white")
  | link(f, r) =>
    above(rectangle(20, 20, "solid", f),
      stack(r))
end
```

```
cases(List) str-lst:
  | empty => 0
  | link(fst,rst) => 1 + len(rst)
end
```

```
cases(List) str-list:
  | empty => false
  | link(fst, rst) =>
    if fst == to-find:
      true
    else:
      is-member(rst, to-find)
    end
end
```

```
cases(List) nums:
  | empty => empty
  | link(fst, rst) =>
    if num-round(fst) == fst:
      link(fst, whole-nos(rst))
    else:
      whole-nos(rst)
    end
end
```