```
 1: #!/afs/cats.ucsc.edu/courses/cmps112-wm/usr/smalltalk/bin/gst -f
 2: "$Id: tree.st,v 1.2 2019-02-11 13:16:46-08 - - $"
 3:
 4: nl := Character nl.
 5:
 6: Object subclass: Leaf [
 7:    |char count|
 8:    char [ ^ char ]
 9:    count [ ^ count ]
10:
11:    Leaf class >> new [
12:       self shouldNotImplement
13:    ]
14:
15:    Leaf class >> new: aChar count: aCount [
16:       |result|
17:       result := super new.
18:       result setChar: aChar andCount: aCount.
19:       ^result
20:    ]
21:
22:    setChar: aChar andCount: aCount [
23:       char := aChar.
24:       count := aCount.
25:    ]
26:
27:    <= other [
28:       ^ (count < other count)
29:       | ((count = other count) & (char <= other char))
30:    ]
31:
32:    printBase: aStream [
33:       ^ aStream << self class << '(' << char << ',' << count
34:    ]
35:
36:    printOn: aStream [
37:       (self printBase: aStream) << ')'.
38:    ]
39:
40:    depthFirst: visitor prefix: string [
41:       visitor value: char value: string.
42:    ]
43:
44: ]
45:
```

```
46:
47: Leaf subclass: Tree [
48:    |left right|
49:
50:    Tree class >> new: aChar count: aCount [
51:       self shouldNotImplement
52:    ]
53:
54:    Tree class >> new: aChar count: aCount left: aLeft right: aRight [
55:       |result|
56:       result := super new: aChar count: aCount.
57:       result setLeft: aLeft andRight: aRight.
58:       ^ result
59:    ]
60:
61:    setLeft: aLeft andRight: aRight [
62:       left := aLeft.
63:       right := aRight.
64:    ]
65:
66:    printOn: aStream [
67:       (self printBase: aStream) << ',' << left << ',' << right << ')'.
68:    ]
69:
70:    depthFirst: visitor prefix: string [
71:       left depthFirst: visitor prefix: string, '0'.
72:       right depthFirst: visitor prefix: string, '1'.
73:    ]
74:
75: ]
76:
```

```
 77:
 78: a := Leaf new: $a count: 10.
 79: b := Leaf new: $b count: 20.
 80: c := Leaf new: $c count: 15.
 81: t := Tree new: $t count: 30 left: a right: b.
 82: u := Tree new: $u count: 50 left: t right: c.
 83: x := Leaf new: $x count: 20.
 84: z := Tree new: $z count: 80 left: u right: x.
 85:
 86: sortcol := SortedCollection new.
 87: sortcol add: t; add: u; add: a; add: b; add: x; add: z; inspect.
 88:
 89: stdout << nl << 'Before vising z Tree' << nl.
 90: z depthFirst: [:char :string |
 91:    stdout << '[' << char << ']=' << string << nl.
 92: ] prefix: ''.
 93:
 94: stdout << nl << 'Before sortcol do: loop' << nl.
 95: sortcol do: [:item |
 96:    stdout << item << nl.
 97: ].
 98:
 99: stdout << nl << 'Before remove loop.' << nl.
100: [sortcol notEmpty] whileTrue: [
101:    |first|
102:    first := sortcol removeFirst.
103:    stdout << first << nl.
104: ]
105:
106: "TEST: tree.st"
107:
```

```
    1: ::::::::::::::::::::::::::::::
    2: mkst: tree.st
    3: ::::::::::::::::::::::::::::::
    4: An instance of SortedCollection
    5:   firstIndex: 1
    6:   lastIndex: 6
    7:   lastOrdered: 6
    8:   sorted: true
    9:   sortBlock: a BlockClosure
   10:   contents: [
   11:     [1]: Leaf(a,10)
   12:     [2]: Leaf(b,20)
   13:     [3]: Leaf(x,20)
   14:     [4]: Tree(t,30,Leaf(a,10),Leaf(b,20))
   15:     [5]: Tree(u,50,Tree(t,30,Leaf(a,10),Leaf(b,20)),Leaf(c,15))
   16:     [6]: Tree(z,80,Tree(u,50,Tree(t,30,Leaf(a,10),Leaf(b,20)),Leaf(c,15)
),Leaf(x,20))
   17:   ]
   18:
   19: Before vising z Tree
   20: [a]=000
   21: [b]=001
   22: [c]=01
   23: [x]=1
   24:
   25: Before sortcol do: loop
   26: Leaf(a,10)
   27: Leaf(b,20)
   28: Leaf(x,20)
   29: Tree(t,30,Leaf(a,10),Leaf(b,20))
   30: Tree(u,50,Tree(t,30,Leaf(a,10),Leaf(b,20)),Leaf(c,15))
   31: Tree(z,80,Tree(u,50,Tree(t,30,Leaf(a,10),Leaf(b,20)),Leaf(c,15)),Leaf(x,
20))
   32:
   33: Before remove loop.
   34: Leaf(a,10)
   35: Leaf(b,20)
   36: Leaf(x,20)
   37: Tree(t,30,Leaf(a,10),Leaf(b,20))
   38: Tree(u,50,Tree(t,30,Leaf(a,10),Leaf(b,20)),Leaf(c,15))
   39: Tree(z,80,Tree(u,50,Tree(t,30,Leaf(a,10),Leaf(b,20)),Leaf(c,15)),Leaf(x,
20))
   40: ::::::::::Exit status 0
```