```
 1: #!/afs/cats.ucsc.edu/courses/cmps112-wm/usr/smalltalk/bin/gst -f
 2: "$Id: tree.st,v 1.10 2012-02-23 18:57:51-08 - - $"
 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:     inorder: 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:    inorder: visitor prefix: string [
71:       left inorder: visitor prefix: string, '0'.
72:       right inorder: 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 inorder: [: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: ****mks**** tree.st 2>&1
 2: An instance of SortedCollection
 3:    firstIndex: 1
 4:    lastIndex: 6
 5:    lastOrdered: 6
 6:    sorted: true
 7:    sortBlock: a BlockClosure
 8:    contents: [
 9:       [1]: Leaf(a,10)
10:       [2]: Leaf(b,20)
11:       [3]: Leaf(x,20)
12:       [4]: Tree(t,30,Leaf(a,10),Leaf(b,20))
13:       [5]: Tree(u,50,Tree(t,30,Leaf(a,10),Leaf(b,20)),Leaf(c,15))
14:       [6]: Tree(z,80,Tree(u,50,Tree(t,30,Leaf(a,10),Leaf(b,20)),Leaf(c,15)
),Leaf(x,20))
15:    ]
16:
17: Before vising z Tree
18: [a]=000
19: [b]=001
20: [c]=01
21: [x]=1
22:
23: Before sortcol do: loop
24: Leaf(a,10)
25: Leaf(b,20)
26: Leaf(x,20)
27: Tree(t,30,Leaf(a,10),Leaf(b,20))
28: Tree(u,50,Tree(t,30,Leaf(a,10),Leaf(b,20)),Leaf(c,15))
29: Tree(z,80,Tree(u,50,Tree(t,30,Leaf(a,10),Leaf(b,20)),Leaf(c,15)),Leaf(x,
20))
30:
31: Before remove loop.
32: Leaf(a,10)
33: Leaf(b,20)
34: Leaf(x,20)
35: Tree(t,30,Leaf(a,10),Leaf(b,20))
36: Tree(u,50,Tree(t,30,Leaf(a,10),Leaf(b,20)),Leaf(c,15))
37: Tree(z,80,Tree(u,50,Tree(t,30,Leaf(a,10),Leaf(b,20)),Leaf(c,15)),Leaf(x,
20))
38: ****status: 0x0000
39:
```