CSc 451, Spring 2003
Assignment 4
Due: Thursday, February 13 at 18:00

Problem 1. (7 points)

Write a procedure `randlist(N, p)` that generates lists with lengths ranging randomly from 1 through N. Each element of the list is either an integer from 1 through 100 (randomly chosen) or another random list having the same potential characteristics. The argument `p` represents the percentage, on average, of list elements that are lists.

Examples:

```
][ randlist(5,30);
   r := L1:[55,L2:[60,64,17],88,L3:[6,9,83,99,8]]  (list)

][ randlist(5,30);
   r := L1:[88,19,95]  (list)

][ randlist(5,30);
   r := L1:[57,55,23,L2:[L3:[36,L4:[96,10,L5:[36,L6:[9,28,
59],L7:[2,L8:[L9:[39],59,11,L10:[L11:[L12:[61,2,36],78],67,6
4,L13:[52],19]]],48],L14:[L15:[33],L16:[39]]],50]],L17:[L18:
[44,81,36,59],93,18,20]]  (list)

][ randlist(5,30);
   r := L1:[40,L2:[57],84,L3:[77],53]  (list)
```

Note that `randlist` is a generator:

```
][ .every randlist(3,20) \ 10;
   L1:[84,54]  (list)
   L2:[7,63]  (list)
   L3:[69]  (list)
   L4:[14]  (list)
   L5:[L6:[81,36,34],L7:[32,25]]  (list)
   L8:[18,91,L9:[38,31,49]]  (list)
   L10:[6]  (list)
   L11:[5,L12:[L13:[44,84]],79]  (list)
   L14:[94]  (list)
   L15:[88,12]  (list)
```

Assume `N >=0` and `0 <= p <= 100`.

Note that various combinations of `N` and `p` produce varying likelihoods of sufficient recursion to overflow the stack. For example, `randlist(1,100)` is sure to exhaust the stack, but `randlist(10,5)` is not likely to.

Problem 2. (7 points)

Write a procedure `ltos(L)` that produces a string representation of the list `L`.

```
][ ltos([3,2,1,5]);
   r := "[3,2,1,5]"  (string)

][ ltos([3,2,[10,[],[],5]]);
   r := "[3,2,[10,[],[],5]]"  (string)

][ ltos([[[]]]);
   r := "[[[]]]"  (string)

][ randlist(5,30);
   r := L1:[L2:[L3:[L4:[63,71],24,32,90],2,35,42],98] (list)

][ ltos(r);
   r := "[[[[63,71],24,32,90],2,35,42],98]"  (string)

][ every write(ltos(randlist(3, 40))) \ 10;
[32]
[31,[[[63,71],24,32],[2,35,42],98]]
[59,15]
[49,[62,18,74],95]
[52,[82,[27,59],[[[42,89,[[69]]],20],50,43]]]
[[71],79]
[8,[21],12]
[[48],85]
[51,[54,65,[31,69,[[[81,36,34],[32,25]],83,[5,54]]]]]
[[49],[[[[44,84]]],79,[37,[12,[14,[47]],6],33]],67,8]]
Failure
```

Assume that `L` is not cyclic and contains only integers and other lists, which may in turn contain lists of integers and lists. **Restriction: You may not use `Image()`.**

Miscellaneous

No comments or explanation of any sort need be included with your solutions.

You are specifically prohibited from directly copying any code, except that presented in class or otherwise provided by me. However, you may study discovered code, such as that found in a textbook—not the code of a classmate—to the point of understanding how it works and then with that knowledge, write your own version.

Deliverables

Use `turnin` with the tag `451_4` to submit your solutions for grading. The only deliverable for this assignment is the file `randlist.icn`. It should contain the procedures `randlist` and `ltos`, and any helper procedures that are required. It should NOT contain a main procedure.

Reference Versions

Reference versions of `randlist` and `ltos` are in the ucode file pair `ref_randlist.u1` and `ref_randlist.u2` in `/home/cs451/a4`. They are linked into a driver program named `rmain`, which takes three command line arguments: The number of lists to generate, the list length, and the sublist probability.

Here is `rmain` in operation:

```
% /home/cs451/a4/rmain 3 5 15
[32,43]
[[[[63,71],24,32,90],2,35,42],98]
[59,15]

% /home/cs451/a4/rmain 3 5 15
[32,43]
[[[[63,71],24,32,90],2,35,42],98]
[59,15]

% /home/cs451/a4/rmain -s 3 5 15
[[[75,72,70,[6]],63],96,30]
[1,98,9,40,2]
[44,92,92]

% /home/cs451/a4/rmain -s 3 5 15
[65]
[37]
[17,[[83,79],8],73,47,100]
```

Note that without the `-s` flag, `rmain` produces the same results every time for a given set of parameters.

Here is a trivial main program (`rsimple.icn`) that shows how to use `link` to access the reference version:

```
link "/home/cs451/a4/ref_randlist"

procedure main()
    every write(ltos(randlist(5, 20))) \ 5
end
```

Compile with `icont rsimple.icn` to produce an executable.

To link in your version of `randlist`, comment or remove the `link` directive and compile with `icont rsimple.icn randlist.icn`.

If you want to experiment with deep-diving combinations you should increase the size of Icon's main interpreter stack by setting the environment variable `MSTKSIZE`. The default is 10,000.