Problem-based Intro. to Computer Science 2011-1
Duck Duck Goose Elimination Lab 7

10/24/2011

Image from www.inkape.com

1 Problem

'Duck, Duck, Goose' is a children's game played in many cultures. To play the game, the children sit in a circle, and the first child in the circle is chosen to be 'IT', removing 'IT' from the circle. The player who is 'IT' runs around the circle calling out 'Duck!' as they pass each player and eventually picks one of the children in the circle to be the 'Goose'. The 'Goose' then stands up and races the 'IT' player around the circle until one of them reaches the empty seat where the goose was sitting. The loser of the round is eliminated from the game. The winner of the round takes the empty seat. The child after the winner becomes 'IT' for the next round. The game continues this way until there is only one child left, the winner. Here is a text file, `test.txt`, that your program might read:

```
Tim

Sam

Pat

Jim
```

Here is output from using the above text file:

```
Enter a file for names of the players: test.txt
Enter a seed for the random number generator: 1577
Playing the game...
================================================================
START Round  1  players are: <Tim,Sam,Pat,Jim>
Tim is IT.
Sam is the goose!

Goose Sam lost!
Goose Sam leaves the game!
END Round  1  players are: <Pat,Jim,Tim>
================================================================
START Round  2  players are: <Pat,Jim,Tim>
Pat is IT.
Jim is a duck.
Tim is a duck.
Jim is a duck.
Tim is the goose!

Goose Tim wins!
IT Pat leaves the game!
END Round  2  players are: <Jim,Tim>
================================================================
START Round  3  players are: <Jim,Tim>
Jim is IT.
Tim is a duck.
Tim is a duck.
Tim is the goose!

Goose Tim wins!
IT Jim leaves the game!
END Round  3  players are: <Tim>
```

1.1 Problem-solving Session (20%)

Students will work in teams of four to five students for problem-solving.

Your *players* will be *the members in your problem-solving team*. Make your names unique.

1. Simulate the game, picking someone who is 'IT' to play a few rounds of elimination. Be sure to simulate 'IT' both winning and losing rounds, as well as rounds that circle around more than once.
2. Design a linked-node-based data structure that will represent the simulation. You may not use Python's built-in lists.
3. Draw a picture showing how the data structure behaves as the game is played. Consider how 'IT' moves around the circle and how the list changes as 'IT' wins or loses.
4. Identify the operations the list data structure will need to support in order to enable you to implement this game. Define the operation's signatures with:
   - function name,
   - parameters,
   - return type/value,
   - preconditions, and postconditions.

   (You will create a list module containing these list structures and functions.)
5. Write pseudocode for the game's `play` function using the identified list operations.

Hand in your problem-solving materials when the instructor requests them. Some teams in the class then will have a chance to present their results.

1.2 Implementation Overview (80%)

You will complete this portion of the lab on your own. Another handout will provide more detailed requirements. *You may not use Python's built-in lists or any other built-in collections for this assignment.*

Summary of Tasks:

- Write a `myList.py` *library module* for the program to import.
- Write a `ddg.py` program that prompts for information and plays the game.
- Write a `report.txt` file providing the results of running tests on your Duck Duck Goose program.

Handout 2 for the start of the Lab Session

1.3 Implementation Requirements

Use a random number to determine whether or not the current location is a duck or the goose as 'IT' proceeds. The Python `random` module has a random number module you can import. After seeding the generator, a call to `random.random()` returns a floating point number in the range [0.0-1.0). See the pseudocode later in this document. If you seed the random number generator by calling `random.seed(intSeed)` with the same value before calling `random.random()`, then the output sequence of random numbers will be the same. This is helpful for debugging your program and checking to see if you get the same output as the example output shown.

Design your program so that there is a 75% chance that any given player is the duck (or a 25% chance of a goose), and a 50% chance that the goose or 'IT' wins the round.

When each round of the program starts, it prints the list of children. The first child in the list will always be the one who becomes 'IT'.

1.4 The myList Interface

While there are other operations your list implementation may need, the following are the functions you *must implement*:

1. `mkList( )` : creates a new, empty list instance
2. `printList( lst )` : print the elements in the list instance
3. `add( lst, element )` : adds the element to the list
4. `advance( lst )` : advances to the next position in the list
5. `remove( lst )` : removes the element at the next position
6. `get( lst )` : returns the element at the next position
7. `size( lst )` : return the size of the list

(Optional: you might want to implement a `set( lst, element )` function.)

1.5 Outline of the main program

The main program must use random numbers to decide who wins or loses. To do so, the module must import the library, seed the random generator, and then generate random numbers for two different purposes: to decide when or whether to keep marching around the circle, and to decide whether the goose or 'IT' wins. The following outline shows the overall flow of the main program.

```
import  random

...
    seed = int( input( "Enter seed for the random number generator: " ) )
    random.seed( seed )    # set the seed
    ...
```

```
add the players from the file.
...
loop until only one child is left:   # start of a round
    ...
    choose IT and remove IT from the circle.
    ...
    while random.random() < 0.75: # a random number of times
        ...
        walk to the next position in the circle

    IT picks the child at the next position as the goose.
    ...
    if random.random() < 0.5:
        The goose wins, and IT is out of the game.
        The goose stays in its place.
    else:
        IT wins, and the goose is out of the game.
        IT is in goose's old place.
    print the players at the end of this round.
end of game
```

1.6 Submission

The files for testing are `names1.txt`, `names2.txt`, and `names3.txt`; your instructor should have made these available through MyCourses.

Submit the following materials:

1. 10%: **report.txt**: This text file should **identify the winner** for each supplied input file. Use this random number: 9339. **Make sure to cast it from string to int!**
2. 40%: **myList.py**: The linked list module implementation which should have test functions for the linked list functions.
3. 25%: **ddg.py**: The program file for the game of 'Duck, Duck, Goose' imports the `myList.py` library module, asks the user for the file name, then the random number seed value, and plays the game.
4. 5%: Style covers documentation of authorship, docstrings with function type specification, purpose statement, and the requisite pre-conditions or post-conditions.

When you have completed the assignment, put your files in **labduck.zip** and submit the zip file to the dropbox for this assignment.