TCSS 342 - Data Structures
Extra Assignment 2 - Evolved Names

Time Estimate: 8-12 hours

The purpose of this assignment is to:

- Use your linked list data structure.
- Rely on your sorting procedure for a linked list.
- Follow the specification document exactly.
- Practice programming and debugging in randomized environments.
- Explore evolutionary algorithms.

Overview

Imagine a virtual world in which all that exists are strings of characters from this set

{ A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, _, -, ' }

(The '_' represents the space character.) Strings in this world can reproduce new strings and die if they are not fit enough. You will evolve strings in this world until they spell your name. To do this you will create a Genome class which will contain a list of characters from the above set representing a string in your world, and you will create a Population class which will contain a list of Genomes representing all the strings in your world. Using the principles of evolution you will allow this population to evolve over generations until one of the strings reaches the target.

To complete this you will implement two public classes:

- Genome
- Population

Formal Specifications

Population

`+population : MyLinkedList<Genome>`
`+mostFit : Genome`
`+generation : int`
`-size : int`

`+nextGeneration()` `+toString() : String`

Field summary

- `population` - This list stores the Genomes in the population. The initial population of

size `size` is created in the constructor. The population is updated every time `nextGeneration` is called.

- `mostFit` - This stores the most fit genome, that is, the genome with maximum fitness. After sorting the population by fitness in `nextGeneration` this value is extracted from the list.
- `generation` - The number of times `nextGeneration` has been called.
- `size` - The number of genomes in the `population`. For this experiment we'll use a population size of 100.

Method summary

- `nextGeneration` - This function is called to update the population to the next generation. To do that carry out these steps:
    - Delete the half of the population with the lowest fitness. The remaining members of the population will serve as possible parents of new members.
    - Create new genomes from the remaining population until the population is restored to its original size by doing either of the following with equal chance:
        - Pick a remaining genome at random and clone it (with the copy constructor). Then mutate the clone.
        - Pick a remaining genome at random and clone it. Then crossover the clone with another remaining genome (selected at random). Then mutate the result.
    - Sort the population by fitness.
    - Update the `mostFit` variable to the genome with the highest fitness in the population.

Genome implements Comparable<Genome>
#genes : MyLinkedList<Character> -target : MyLinkedList<Character> -mutationRate : double
+Genome(genome : Genome) +mutate() +crossover(parent : Genome) +fitness() : int +compareTo(other : Genome) : int +toString() : String

Field summary

- `genes` - A list of characters representing the string encoded by the genome. This list is either copied from another genome (using the copy constructor) or initialized to the empty list.
- `target` - The target string. This variable is initialized in the constructor and never modified. Initialize this to your name, or to “CHRISTOPHER PAUL MARRIOTT” to test it against my name. Note that the longer the target string, the longer it takes evolution to find it.
- `mutationRate` - The mutation rate for the population. For this experiment we will use a mutation rate of 0.05.

Method summary

- `Genome` - You should have two different constructors.
    - The default constructor takes no arguments and initializes genes to the empty list.
    - The copy constructor takes a single `genome` as an argument copies `genes` from the `genome`.
- `mutate` - This method is called to mutate the genes. We will use the following rules (each rule applies every time mutate is called):
    - With `mutationRate` chance add a randomly selected character to a randomly selected position in genes.
    - With `mutationRate` chance delete a single character from a randomly selected position of genes but do this only if its length is at least 1.
    - For each character in genes:
        - with `mutationRate` chance the character is replaced by a randomly selected character.
- `crossover` - This function will update its genes by crossing it over with the `parent`. Create the new genes list by following these steps for each position in the list:
    - Randomly choose one of the two parent lists.
    - If the parent list has a character at this position (i.e. it is long enough) copy that character into the new list. Otherwise end the new list here.
- `fitness` - Returns the fitness calculated using the following formula:

$$-(l + d)$$

where l is the difference in length between genes and the target and d is the number of characters that are incorrect (a missing character or an extra character are also considered mismatches).

Example: The initial string “” has no characters and my name “CHRISTOPHER PAUL

MARRIOTT" has 25 characters. The difference in length $l = 25$ and the number of mismatches $d = 25$ since there are 25 missing characters. So the total fitness of "" is -50.

- `compareTo` - Compares the two genomes based on their fitness.
- `toString` - Outputs the genome and its fitness in this format:

(genes, fitness)

Testing

It is a good idea to test your Genome class before placing it into the evolutionary simulation. In particular, test the mutate, crossover and fitness to make sure they have the appropriate behavior and output. When testing the mutate function it can be helpful to set your mutation rate to 1 and test each of the rules independently.

My main function is very simple. Other than output and timer code this is all I have:

```
while(pop.mostFit.fitness() < 0) {
        System.out.print("Generation " + pop.generation + " ");
        System.out.println(pop.mostFit);
        pop.nextGeneration();
}
```

Here is the output from my solution:

```
Evolving names...
Generation 0 ("", -50)
Generation 1 ("", -50)
Generation 2 ("E", -49)
Generation 3 ("DO", -48)
Generation 4 ("DO", -48)
Generation 5 ("DO", -48)
Generation 6 ("WH", -47)
Generation 7 ("WH", -47)
Generation 8 ("AZR", -46)
Generation 9 ("AZR", -46)
Generation 10 ("AZR", -46)
Generation 11 ("WHR", -45)
Generation 12 ("WHR", -45)
Generation 13 ("AHRT", -44)
Generation 14 ("AHRT", -44)
Generation 15 ("AHRT", -44)
Generation 16 ("AHRT", -44)
Generation 17 ("AHRT", -44)
Generation 18 ("AHRRU", -43)
```

Generation 19 ("AHRRU", -43)
Generation 20 ("AHRRU", -43)
Generation 21 ("AHRRU", -43)
Generation 22 ("AHRTTP", -42)
Generation 23 ("AHRTTP", -42)
Generation 24 ("AHRTTP", -42)
Generation 25 ("WHRFPPQ", -41)
Generation 26 ("WHRFPPQ", -41)
Generation 27 ("CHRLSU", -40)
Generation 28 ("CHRLSU", -40)
Generation 29 ("CHRLSU", -40)
Generation 30 ("CHRLSU", -40)
Generation 31 ("WHRIEQO", -39)
Generation 32 ("CHRPSZAQ", -38)
Generation 33 ("CHRPSZAQ", -38)
Generation 34 ("CHRPSZAQ", -38)
Generation 35 ("CHRPSZAQ", -38)
Generation 36 ("WHRTSGOP", -37)
Generation 37 ("CHRHSZOP", -36)
Generation 38 ("CHRHSZOP", -36)
Generation 39 ("CHRHSZOP", -36)
Generation 40 ("CHRHSZOP", -36)
Generation 41 ("CHRHSZOP", -36)
Generation 42 ("CHR'SBOPQ", -35)
Generation 43 ("CHRISPOPL", -34)
Generation 44 ("CHRISPOPL", -34)
Generation 45 ("CHRISPOPL", -34)
Generation 46 ("CHRISPOPL", -34)
Generation 47 ("CHRISPOPL", -34)
Generation 48 ("CHRISPOPL", -34)
Generation 49 ("CHRISPOPL", -34)
Generation 50 ("CHRISPOPLN", -33)
Generation 51 ("CHRISPOPLN", -33)
Generation 52 ("CHRISPOPLN", -33)
Generation 53 ("CHRISPOPLE", -32)
Generation 54 ("CHRISPOPLE", -32)
Generation 55 ("CHRISPOPLE", -32)
Generation 56 ("CHRISPOPLE", -32)
Generation 57 ("CHRISPOPLE", -32)
Generation 58 ("CHRISTOPLE", -31)
Generation 59 ("CHRISTOPLE", -31)
Generation 60 ("CHRISTOPLE", -31)
Generation 61 ("CHRISTOPLE", -31)

Generation 62 ("CHRISTOPHE", -30)
Generation 63 ("CHRISTOPHE", -30)
Generation 64 ("CHRISTOPHEJ", -29)
Generation 65 ("CHRISTOPHEJ", -29)
Generation 66 ("CHRISTOPHEJ", -29)
Generation 67 ("CHRISTOPHEJ", -29)
Generation 68 ("CHRISTOPHEJ", -29)
Generation 69 ("CHRISTOPHEJ", -29)
Generation 70 ("CHRISTOPHEW-", -28)
Generation 71 ("CHRISTOPHEW-", -28)
Generation 72 ("CHRISTOPHEW-", -28)
Generation 73 ("CHRISTOPHEW-", -28)
Generation 74 ("CHRISTOPHEW-", -28)
Generation 75 ("CHRISTOPHEW-", -28)
Generation 76 ("CHRISTOPHEW-", -28)
Generation 77 ("CHRISTOPHEW-", -28)
Generation 78 ("CHRISTOPHEW-", -28)
Generation 79 ("CHRISTOPHEW-", -28)
Generation 80 ("CHRISTOPHEW-", -28)
Generation 81 ("CHRISTOPHEW-E", -27)
Generation 82 ("CHRISTOPHEW-E", -27)
Generation 83 ("CHRISTOPHEW-E", -27)
Generation 84 ("CHRISTOPHEW-E", -27)
Generation 85 ("CHRISTOPHEW-E", -27)
Generation 86 ("CHRISTOPHEW-E", -27)
Generation 87 ("CHRISTOPHEW-E", -27)
Generation 88 ("CHRISTOPHEW-E", -27)
Generation 89 ("CHRISTOPHEW-E", -27)
Generation 90 ("CHRISTOPHEW-E", -27)
Generation 91 ("CHRISTOPHEK EE", -25)
Generation 92 ("CHRISTOPHEK EE", -25)
Generation 93 ("CHRISTOPHEK EE", -25)
Generation 94 ("CHRISTOPHEK EE", -25)
Generation 95 ("CHRISTOPHEK EE", -25)
Generation 96 ("CHRISTOPHEK EE", -25)
Generation 97 ("CHRISTOPHEK EE", -25)
Generation 98 ("CHRISTOPHEK EE", -25)
Generation 99 ("CHRISTOPHEK EE", -25)
Generation 100 ("CHRISTOPHEK EE", -25)
Generation 101 ("CHRISTOPHE EET", -24)
Generation 102 ("CHRISTOPHE REET", -23)
Generation 103 ("CHRISTOPHE REET", -23)
Generation 104 ("CHRISTOPHE REET", -23)

Generation 105 ("CHRISTOPHE REET", -23)
Generation 106 ("CHRISTOPHE REET", -23)
Generation 107 ("CHRISTOPHE CREET", -22)
Generation 108 ("CHRISTOPHE CREET", -22)
Generation 109 ("CHRISTOPHE CREET", -22)
Generation 110 ("CHRISTOPHE CREET", -22)
Generation 111 ("CHRISTOPHEK P'EED", -21)
Generation 112 ("CHRISTOPHEK P'EED", -21)
Generation 113 ("CHRISTOPHEK P'EED", -21)
Generation 114 ("CHRISTOPHEK P'EED", -21)
Generation 115 ("CHRISTOPHEK P'EED", -21)
Generation 116 ("CHRISTOPHER EEUQA", -20)
Generation 117 ("CHRISTOPHER EEUQA", -20)
Generation 118 ("CHRISTOPHER EEUQA", -20)
Generation 119 ("CHRISTOPHEQ KALLGI", -19)
Generation 120 ("CHRISTOPHEQ KALLGI", -19)
Generation 121 ("CHRISTOPHEQ KALLGI", -19)
Generation 122 ("CHRISTOPHEQ KALLGI", -19)
Generation 123 ("CHRISTOPHER PEULT", -18)
Generation 124 ("CHRISTOPHER PEULT", -18)
Generation 125 ("CHRISTOPHER PEULT", -18)
Generation 126 ("CHRISTOPHER KAULGA", -17)
Generation 127 ("CHRISTOPHER KAULGA", -17)
Generation 128 ("CHRISTOPHER PLULDAC", -16)
Generation 129 ("CHRISTOPHER PLULDAC", -16)
Generation 130 ("CHRISTOPHER PLULDAC", -16)
Generation 131 ("CHRISTOPHER PAULXAI", -15)
Generation 132 ("CHRISTOPHER PAULXAI", -15)
Generation 133 ("CHRISTOPHER PAULXAI", -15)
Generation 134 ("CHRISTOPHER PAULXAI", -15)
Generation 135 ("CHRISTOPHER PAULXAI", -15)
Generation 136 ("CHRISTOPHER PAULXAI", -15)
Generation 137 ("CHRISTOPHER PAULXAI", -15)
Generation 138 ("CHRISTOPHER PAULSIA", -14)
Generation 139 ("CHRISTOPHER PAULSIA", -14)
Generation 140 ("CHRISTOPHER PAULSIA", -14)
Generation 141 ("CHRISTOPHER PAULSIA", -14)
Generation 142 ("CHRISTOPHER PAULSIA", -14)
Generation 143 ("CHRISTOPHER PAUHWRAI'", -13)
Generation 144 ("CHRISTOPHER PAUHWRAI'", -13)
Generation 145 ("CHRISTOPHER PAUHWRAI'", -13)
Generation 146 ("CHRISTOPHER PAUHWRAI'", -13)
Generation 147 ("CHRISTOPHER PAUL IDI'", -12)

Generation 148 ("CHRISTOPHER PAUL IDI'", -12)
Generation 149 ("CHRISTOPHER PAUL IDI'", -12)
Generation 150 ("CHRISTOPHER PAUL DAI'", -11)
Generation 151 ("CHRISTOPHER PAUL DAI'", -11)
Generation 152 ("CHRISTOPHER PAUL DAI'", -11)
Generation 153 ("CHRISTOPHER PAUL DAI'", -11)
Generation 154 ("CHRISTOPHER PAUL DAI'", -11)
Generation 155 ("CHRISTOPHER PAUL DAI'", -11)
Generation 156 ("CHRISTOPHER PAUL DAI'", -11)
Generation 157 ("CHRISTOPHER PAUL DAI'", -11)
Generation 158 ("CHRISTOPHER PAUL DAI'", -11)
Generation 159 ("CHRISTOPHER PAUL DAI'", -11)
Generation 160 ("CHRISTOPHER PAUL DAI'", -11)
Generation 161 ("CHRISTOPHER PAUL DAI'", -11)
Generation 162 ("CHRISTOPHER PAUL DAI'", -11)
Generation 163 ("CHRISTOPHER PAUL RARF", -10)
Generation 164 ("CHRISTOPHER PAUL RARF", -10)
Generation 165 ("CHRISTOPHER PAUL RARF", -10)
Generation 166 ("CHRISTOPHER PAUL RARF", -10)
Generation 167 ("CHRISTOPHER PAUL RARF", -10)
Generation 168 ("CHRISTOPHER PAUL RARF", -10)
Generation 169 ("CHRISTOPHER PAUL RARF", -10)
Generation 170 ("CHRISTOPHER PAUL QAR'", -9)
Generation 171 ("CHRISTOPHER PAUL QAR'", -9)
Generation 172 ("CHRISTOPHER PAUL QAR'", -9)
Generation 173 ("CHRISTOPHER PAUL QAR'", -9)
Generation 174 ("CHRISTOPHER PAUL QAR'", -9)
Generation 175 ("CHRISTOPHER PAUL QAR'", -9)
Generation 176 ("CHRISTOPHER PAUL QAR'", -9)
Generation 177 ("CHRISTOPHER PAUL QAR'", -9)
Generation 178 ("CHRISTOPHER PAUL QAR'", -9)
Generation 179 ("CHRISTOPHER PAUL 'ARRS", -8)
Generation 180 ("CHRISTOPHER PAUL 'ARRS", -8)
Generation 181 ("CHRISTOPHER PAUL 'ARRS", -8)
Generation 182 ("CHRISTOPHER PAUL 'ARRS", -8)
Generation 183 ("CHRISTOPHER PAUL 'ARRS", -8)
Generation 184 ("CHRISTOPHER PAUL 'ARRS", -8)
Generation 185 ("CHRISTOPHER PAUL 'ARRS", -8)
Generation 186 ("CHRISTOPHER PAUL 'ARRS", -8)
Generation 187 ("CHRISTOPHER PAUL 'ARRS", -8)
Generation 188 ("CHRISTOPHER PAUL RARRK'", -7)
Generation 189 ("CHRISTOPHER PAUL MARRK'", -6)
Generation 190 ("CHRISTOPHER PAUL MARRK'", -6)

Generation 191 ("CHRISTOPHER PAUL MARRK'", -6)
Generation 192 ("CHRISTOPHER PAUL MARRK'", -6)
Generation 193 ("CHRISTOPHER PAUL MARRK'", -6)
Generation 194 ("CHRISTOPHER PAUL MARRK'", -6)
Generation 195 ("CHRISTOPHER PAUL MARRK'", -6)
Generation 196 ("CHRISTOPHER PAUL MARRK'", -6)
Generation 197 ("CHRISTOPHER PAUL MARRK'", -6)
Generation 198 ("CHRISTOPHER PAUL MARRK'", -6)
Generation 199 ("CHRISTOPHER PAUL MARRK'", -6)
Generation 200 ("CHRISTOPHER PAUL MARRK'", -6)
Generation 201 ("CHRISTOPHER PAUL MARRKO", -5)
Generation 202 ("CHRISTOPHER PAUL MARRKO", -5)
Generation 203 ("CHRISTOPHER PAUL MARRKO", -5)
Generation 204 ("CHRISTOPHER PAUL MARRKO", -5)
Generation 205 ("CHRISTOPHER PAUL MARRKO", -5)
Generation 206 ("CHRISTOPHER PAUL MARRKO", -5)
Generation 207 ("CHRISTOPHER PAUL MARRKO", -5)
Generation 208 ("CHRISTOPHER PAUL MARRKO", -5)
Generation 209 ("CHRISTOPHER PAUL MARRKO", -5)
Generation 210 ("CHRISTOPHER PAUL MARRKO", -5)
Generation 211 ("CHRISTOPHER PAUL MARRKO", -5)
Generation 212 ("CHRISTOPHER PAUL MARRKO", -5)
Generation 213 ("CHRISTOPHER PAUL MARRKO", -5)
Generation 214 ("CHRISTOPHER PAUL MARRQTKS", -4)
Generation 215 ("CHRISTOPHER PAUL MARRQTKS", -4)
Generation 216 ("CHRISTOPHER PAUL MARRQTKS", -4)
Generation 217 ("CHRISTOPHER PAUL MARRQTKS", -4)
Generation 218 ("CHRISTOPHER PAUL MARRQTKS", -4)
Generation 219 ("CHRISTOPHER PAUL MARRQTKS", -4)
Generation 220 ("CHRISTOPHER PAUL MARRQTKS", -4)
Generation 221 ("CHRISTOPHER PAUL MARRQTKS", -4)
Generation 222 ("CHRISTOPHER PAUL MARRQTKS", -4)
Generation 223 ("CHRISTOPHER PAUL MARRQTKS", -4)
Generation 224 ("CHRISTOPHER PAUL MARRQTKS", -4)
Generation 225 ("CHRISTOPHER PAUL MARRYOKS", -3)
Generation 226 ("CHRISTOPHER PAUL MARRYOKS", -3)
Generation 227 ("CHRISTOPHER PAUL MARRIOKS", -2)
Generation 228 ("CHRISTOPHER PAUL MARRIOKS", -2)
Generation 229 ("CHRISTOPHER PAUL MARRIOKS", -2)
Generation 230 ("CHRISTOPHER PAUL MARRIOKS", -2)
Generation 231 ("CHRISTOPHER PAUL MARRIOKS", -2)
Generation 232 ("CHRISTOPHER PAUL MARRIOKS", -2)
Generation 233 ("CHRISTOPHER PAUL MARRIOKS", -2)

Generation 234 ("CHRISTOPHER PAUL MARRIOKS", -2)
Generation 235 ("CHRISTOPHER PAUL MARRIOKS", -2)
Generation 236 ("CHRISTOPHER PAUL MARRIOKS", -2)
Generation 237 ("CHRISTOPHER PAUL MARRIOKS", -2)
Generation 238 ("CHRISTOPHER PAUL MARRIOKS", -2)
Generation 239 ("CHRISTOPHER PAUL MARRIOKS", -2)
Generation 240 ("CHRISTOPHER PAUL MARRIOKS", -2)
Generation 241 ("CHRISTOPHER PAUL MARRIOKS", -2)
Generation 242 ("CHRISTOPHER PAUL MARRIOYT", -1)
Generation 243 ("CHRISTOPHER PAUL MARRIOYT", -1)
Generation 244 ("CHRISTOPHER PAUL MARRIOYT", -1)
Generation 245 ("CHRISTOPHER PAUL MARRIOYT", -1)
Generation 246 ("CHRISTOPHER PAUL MARRIOYT", -1)
Generation 247 ("CHRISTOPHER PAUL MARRIOYT", -1)
Generation 248 ("CHRISTOPHER PAUL MARRIOYT", -1)
Generation 249 ("CHRISTOPHER PAUL MARRIOYT", -1)
Generation 250 ("CHRISTOPHER PAUL MARRIOYT", -1)
Generation 251 ("CHRISTOPHER PAUL MARRIOYT", -1)
Generation 252 ("CHRISTOPHER PAUL MARRIOYT", -1)
Generation 253 ("CHRISTOPHER PAUL MARRIOYT", -1)
Generation 254 ("CHRISTOPHER PAUL MARRIOTT", 0)
Evolving names took 60 milliseconds.

Submission

You will submit a .zip file containing:

- MyLinkedList.java
- Genome.java
- Population.java

Grading Rubric

In order to count as complete your submission must:

1. Evolve your name in a reasonable amount of time and generations.
    a. My solution typically finishes in under 100ms. You should aim for this.
    b. My solution typically finishes in under 400 generations. You should aim for this.

Reminder: Incomplete assignments can always be corrected and resubmitted. If they are completed within 7 Generations of the due date they will count as late and after that period they will count as missed. Please review the grading matrix for the number of permitted late and missed assignments.