cs3102 - Theory of Computation - s'19 — Nate Brunelle
due monday february 4, 2019 at 11p via Collab

H1.njb2b

Collaboration Policy You are encouraged to collaborate with up to 3 other students, but all work submitted must be your own independently written solution. List the names of all of your collaborators. Do not seek published solutions for any assignments. If you use any published resources when completing this assignment, be sure to cite them. Do not submit a solution that you are unable to explain orally to a member of the course staff.

Submission Format Your submission to collab should be a single zip file containing:

1. all java files required
2. a pdf of your written solutions
3. the tex file used to generate that pdf
4. all other items needed to generate the pdf from the tex file (images, etc.)

problem 1 *Deciders and Enumerators*

As we're exploring a more formal definition of computing this semester, it will be helpful to have a simplified definition of what a computation is. For most of the semester we will be restricting our discussion to that of a "decider". A decider is a computing device that takes a string (i.e. a sequence of characters) as input, and either "accepts" or "rejects" that string (returns true or false respectively). To begin with this assignment, you will be making deciders in Java.

The files "Decider.java", "SigmaStar.java", "PrimeChecker.java", and "IcecreamCone.java" together show examples of deciders. The file "Decider.java" is an interface for deciders, which says that a decider has a "decide" method that takes a String as input and gives a boolean as output. The other three files are implementations of deciders. The "SigmaStar" decider returns true (i.e. accepts) for all strings given (the name is a bit obscure at the moment, but we will discuss the reasoning behind that name soon). The "PrimeChecker" decider returns true on any string whose length is a prime number. The "IcecreamCone" decider returns true on any string that represents a valid ice cream cone (it must start with the cone, then have any number of comma-separated scoops of the available flavors). You don't need to do anything with these examples, they are merely present to give you an idea for some of the diversity of things a decider might do. Each of these has a main method that demonstrates how the decider behaves on various inputs.

Once you are comfortable with how we're implementing these deciders in Java, implement the following on your own (for 10 points each):

- **EvenA.java**: accepts exactly the strings that have an even number of As
- **Palindrome.java**: accepts exactly the strings that are palindromes
- **SumChecker.java**: accepts exactly the strings that represent correct addition expressions of two natural numbers, meaning any string of the form "α+β=γ" where α, β, γ are each strings representing natural numbers (i.e. they contain only the characters 0-9), and when interpreted as numbers $\alpha+\beta=\gamma$ is mathematically correct.

Some examples of accepted and rejected strings for the above are given in the main methods of the corresponding java files. You'll likely want to come up with some of your own to test before submitting your final code.

As we mentioned in class, we can characterize the behavior of a decider by the set of strings which it accepts. A program which produces all of the strings accepted by a decider is called an enumerator. For your final programming task, we're asking you to build an enumerator in Java (for 20 points).

The **Enumerator.java** file should contain this enumerator class. The constructor for this class takes two different parameters. The first is a list of characters that we call the "alphabet". This represents all of the characters that our strings are allowed to have. The second parameter is a decider, and all strings our enumerator will present must be accepted by that decider.

The order that the characters appear in the alphabet is understood to be their "alphabetical order". If I gave the input [a,&,c,7] then 'a' comes before '&' which comes before 'c' which comes before '7' in alphabetical order. If I instead gave [c,a,&,7] then we would understand 'c' as coming first alphabetically.

Your enumerator class must have a method called "enumerate" which takes one parameter. This parameter is an integer representing the number of strings to enumerate. The strings enumerated will be the "lexicographically first" strings accepted by the decider given in the constructor. To order strings lexicographically, they should be ordered first by length (if string s1 is shorter than string s2, s1 always comes first lexicographically, this implies that the empty string is the lexicographically first string), and then alphabetically (if s1 and s2 have the same length, whichever would appear first in the dictionary is lexicographically first).

As an example, if you construct your Enumerator with "Enumerator e = new Enumerator(new char[] {'b','a'}, new SigmaStar())", and then called "e.enumerate(8)" you should receive the array ["", "b", "a", "bb", "ba", "ab", "aa", "bbb"] in exactly that order.

In summary, the programming portion of this assignment requires that you submit 4 .java files (those bolded above):

1. **EvenA.java** (10 points)
2. **Palindrome.java** (10 points)
3. **SumChecker.java** (10 points)
4. **Enumerate.java** (20 points)

Please do not submit any additional .java files.

PROBLEM 2 *Proofs*

Prove each of the following. Be sure to give a formal statement of what you're trying to prove, and mention the technique you're using to prove it.

1. $\sqrt{3} + \sqrt{5}$ is irrational.
2. For any integer x, x^3 is even if and only if x is even.
3. The sum of any four consecutive integers has the form $4k + 2$ for some integer k
4. $n^2 + 5$ is not divisible by 4
5. If a,b, and c are integers and $a^2 + b^2 = c^2$, then at least one of a and b is even.