# Assignment 0

Theodore S. Norvell

Engi 9818 Due 2021 Sept 24.

**Q0 Partitions**

Alice decides to invite 14 friends to an unbirthday party later this month. Unfortunately she discovers that because of their busy schedules she can't have them all over on the same day. In fact she will need to have at least 3 separate unbirthday parties in order that everyone can attend. (Since she has at least 364 unbirthdays a year, this isn't a big problem, but on the other hand she doesn't want to throw more unbirthday parties than necessary, at least not until next month.) So she decides to have 3 unbirthday parties. She'll have to divide her set $S$ of 14 friends into 3 subsets called parts $S_0$, $S_1$, and $S_2$ with the following properties: (a) No one can be missed, so $S_0 \cup S_1 \cup S_2 = S$. And (b) no one should be invited twice, so $S_0 \cap S_1 = \emptyset$, $S_1 \cap S_2 = \emptyset$, and $S_0 \cap S_1 = \emptyset$. Any set of three sets $\{S_0, S_1, S_2\}$ with these two properties is called a partition of $S$ of size 3. We have one more requirement, none of the 3 subsets should be empty – since a party with no guests is hardly a party at all. Alice wonders how many ways she can partition her 14 friends into 3 nonempty subsets.

More generally, we would like to find out how many ways there are to partition a set of size $n$ into $k$ nonempty subsets. I.e., if $S = \{x_0, x_1, ..., x_{n-1}\}$ and the size of $S$ is $n$, a partition into nonempty subsets is a set $\{S_0, S_1, ..., S_{k-1}\}$ in which each part $S_i$ is a subset of $S$, no $S_i$ is empty, the union of all the $S_i$ is $S$ , and for any two subsets $S_i$ and $S_j$, with $i \neq j$, $S_i$ and $S_j$ have no common members ($S_i \cap S_j = \emptyset$).

To check your understanding of this, determine how many ways there are to partitions a set of size 4 into 4, 3, 2, 1, and 0 partitions. You should find there are 1, 6, 7, 1, and 0 ways.

We have the following contract

```
public class Partitions {

    /** Count the number of ways to partition a set of size n into k nonempty subsets.
    *
    * @param n The size of the set
    * @param k The size of the partitions
    * @require 0 <= k and k <= n
    * @ensure result equals the number of ways to partition a set of size n
    */
    public static int countPartitions(int n, int k) {
        // TODO
    }
}
```

Write an implementation of this contract and a set of tests that check that it works.

Hints: First consider the case where $n = 0$ and therefore $k = 0$. There is one partition of $\emptyset$ into 0 subsets, which is $\emptyset$ and all the sets in $\emptyset$ are nonempty, so the answer in this case is 1.

Next consider the case where we have a set $S$ of size $n > 0$ and $k = 0$ . The only set of size 0 is $\emptyset$, and this is not a partition as there must be something in $S$ that is not in any of the members of $\emptyset$, so $\emptyset$ is not a partition of $S$. So the number of partitions in this case is 0.

Next consider the case where $S = \{x_0, x_1, ..., x_{n-1}\}$ is of size $n > 0$ and $k = n$. In this case, there is only one partition of $S$ of size $n$ in which all the members are nonempty, namely $P = \{\{x_0\}, \{x_1\}, ..., \{x_{n-1}\}\}$. So the answer here is 1.

Finally we have the case of $S = \{x_0, x_1, ..., x_{n-1}\}$ with $n > 0$ and $0 < k < n$. Let's break the set of partitions into two disjoint subsets

- We have partitions that contain $\{x_{n-1}\}$. I.e., where $x_{n-1}$ is alone in it's part. (We'll call these partitions the first kind.)
- We have partitions that do not contain $\{x_{n-1}\}$. I.e., where $x_{n-1}$ is not alone in its part. (We'll call these partitions the second kind.)

To check your understanding, consider a set $S = \{a, b, c, d, e\}$ of size 5 with $k = 3$. List all partitions in which $\{e\}$ is a part and then list all partitions where $\{e\}$ is not a part.

Can you use countPartitions to compute the number of partitions of the first kind? (Hint: Consider how these partitions relate to partitions of $\{a, b, c, d\}$ or size 2.)

Can you use countPartitions to compute the number of partitions of the second kind? (Hint: Consider how these partitions relate to partitions of $\{a, b, c, d\}$ of size 3.)

Clearly each partition of is of either the first or second kind and none is of both kinds, so the total number of partitions can be calculated by adding the number of the first kind to the number of the second kind.

**LISP S-Expressions**

The remaining questions all relate to a data type that is used in the language LISP. Called S-Expressions. Each S-Expression is a finite structure that is of exactly one of three kinds.

- Nil is an S-Expression. There is only one Nil.
- There is an infinite set of symbols. There is one symbol for each string, so "hello" is a symbol, "good day" is a symbol and so on. When a symbol is just letters, I won't write the quotes, e.g., hello, unless the symbol is "Nil".
- If $a$ and $b$ are S-Expressions, then $(a.b)$ is an S-Expression. These are called dotted-pairs.

For example (hello . Nil) is an S-Expression and so are all of the following

- ((hello . there) . (all . earthlings))
- ( what . (is . (the . (reason . Nil))))
- Nil
- ( Nil . Nil )

Conventionally S-Expressions are used to represent lists as follows, a list of 4 S-Expressions $(a, b, c, d)$ is represented by an S-Expression

$$(a.(b.(c.(d.\text{Nil}))))$$

and of course this generalizes to lists of all finite lengths. In particular:

- a list of length 0, (), is represented by Nil,
- and if $x$ represents a list of length $n$ then $(a.x)$ represents a list of length $n+1$.

Check your understanding. What S-Expression represents ((a, b, c), (d, e, f))
I've implemented S-Expression in Java. (Download my package from the BrightSpace site.)
The following construction operations are available.

- nil() – return Nil
- symbol( String s ) — returns a symbol
- cons( SExp a, SExp b) — returns a dotted-pair (also called a "cons cell")

Each S-Expression $x$ supports the following methods

- x.isNil() : boolean — true for Nil, false for symbols and dotted-pairs
- x.isAtomic() : boolean — true for Nil and for symbols. false for dotted pairs
- x.eq( SExp y) : boolean — true if both are Nil, true if both are the same symbol, true if both are dotted-pairs created by the same cons operation. Otherwise false.
- x.first() : SExp — if x is a dotted pair $(a.b)$ this returns $a$. Otherwise the operation is not allowed. (It is required that x is not atomic.)
- x.rest() : SExp — if x is a dotted pair $(a.b)$ this returns $b$. Otherwise the operation is not allowed. (It is required that x is not atomic.)

**Q1 Length**
Write a method to compute the length of a list. Write the contract first and then implement the method. Please put your method in a class called A0SExp and use this signature.

```
public static int length(SExp x)
```

Write JUnit tests for this method.

**Q2 Height**
The height of an symbol is 1, the height of Nil is 0 and the height of a dotted $(a.b)$ is 1 more than the height of the highest of $a$ and $b$. Write the contract first and then implement the method. Use this signature.

```
public static int height(SExp x)
```

Write JUnit tests for this method. (Make sure that length would fail at least one of your tests and conversely that height will fail at least on of the test for length.)

**Q3 Lookup**
Suppose we have a list of pairs $((a0.b0), (a1.b1), (a2.b2), ...)$ where all the $a$s are symbols. Write a method that, given a symbol $a$ returns the corresponding $b$. If there is no corresponding $a$, then the result should be Nil. use this signature. Write the contract first and then implement the method. Use this signature

```
public static SExp lookup(Symbol s, SExp list)
```

Write JUnit tests for this method.

**Q4 Replace**

Write a method to replace every occurrence of a symbol $a$ with an S-Expression $x$ in an S-Expression $y$. Write the contract first and then implement the method. Use this signature

```
public static SExp replace(Symbol a, SExp x, SExp y)
```

Write JUnit tests for this method. As an example, the result of replace( k, j, ((k . Nil ). (Nil . (k . z))) ) would be ((j . Nil ). (Nil . (j . z))).

**General instructions.**

For this assignment, all your classes should be in the default package. I.e. in your src folder. For the LISP exercises, you should use these imports

```
import sExpressions.SExp ;
import static sExpressions.SExp.* ;
```

Submission of the assignment will be by Git. More instructions will come about that.

Remember that this is an individual assignment.

The assignment is due on Friday the 24th of Sept.