**Worksheet for Working with Strings**

**Character basics!**

You have to know the building blocks before building something!

In Scheme, in order for the computer to know something is a character it must need “#\” in front of it.

**Practice:**

1. Let’s type (display #\a) in your interaction window and hit enter
2. Can you display character Z in your interaction window?

You can also compare characters!

The syntax for comparing two characters would be

* If they are the same: **(char=? char1 char2)**
* If char1 is before char2 in the alphabet: **(char<? char1 char2)**
* If char1 is either the same or before char2 in the alphabet: **(char<=? char1 char2)**
* If char1 is after char2 in the alphabet: **(char>? char1 char2)**
* If char1 is either the same or after char2 in the alphabet: **(char>=? char1 char2)**

**Let’s try some of them!**

1. Type (char>? #\a #\z) to see if a comes after z in the alphabet or not
2. Can you check to see if character h is before or after i in the alphabet?
3. Let’s check whether if #\a and #\A are the same or not
4. If the answer above is “they are not the same”, then let’s find out if the computer thinks the lower-case alphabet comes before or after the upper-case alphabet
   1. Check if #\a is before or after #\A
   2. Check if #\b is before or after #\a
   3. Last check if #\b is before or after #\A

**String basics!**

Let’s get down to business

We will first learn about how to make strings!

There are a few ways

* If we want a string that is made up of char1 repeated # of times: **(make-string # char1)**
  + Ex: (make-string 5 #\a)
  + Output: “aaaaa”
* A string that contains different characters: **(string char1 char2 char3 char4)**
  + Ex: (string #\a #\p #\p #\l #\e)
  + Output: “apple”
* Another way is that we can define a variable to be a string, so in the definition panel: (define var “string”)
  + Ex. in the definition panel **(define sentence “I love code camp”)**
  + Now if in the interaction panel I type in **sentence**
    - then the output will be: “I love code camp”

**Your turn!**

1. In the interaction panel type in (make-string 10 #\k)
2. In the interaction panel type in (string #\c #\o #\d #\e #\space #\c #\a #\m #\p)
3. In the definition panel type is (define place “Grinnell College”)
   1. Run the program
   2. Then type place in the interaction panel
4. Can you define building as Noyce using (string char1 char2 char3) procedure?

**String manipulation**

* To find the length of a string:(string-length str)
* To find the character in a string at a certain index: (string-ref str #)
* To add two string together: (string-append str1 str2)

Let’s practice a little before moving on to the rest

1. In the interaction panel type in (string-length building)
2. Find the 3rd character in the variable place
   1. Remember we declare place as a variable for a string!
   2. Remember what James said about the index of a string?
3. Combine place and building together using the string-append procedure and declare it as location
   1. The result should be “Grinnell CollegeNoyce”

* You can use this procedure to cut the string into parts: (substring str start# end#)
* Capitalize all characters in a string: (string-upcase str)
* Lowercase all characters in a string: (string-downcase str)
* Capitalize only the first character of each word in a string: (string-titlecase str)

**Practice!**

1. Type in (substring location 0 1)
2. What if I want only the last word in location
3. What if I only want the second and third word the string “I am currently typing”
4. Convert location to all upper-case
5. Convert location to all lower-case

Challenges:

1. Declare the string “today is the second day of the code camp” as second-day. Then change the string so only the first letter of the first word(“today”) is capitalized ("Today") while all the other words stay lower-cased
2. Write a procedure that finds the letter in the middle of a string, if it’s a string of even length then return “there is no middle letter”
3. Write a procedure that capitalizes once every other word

EX: if my input is: “i love code camp”

The output would be: “I love Code camp”

1. Write a simple hangman program

You can enter a letter to the program one at a time. If the letter exists in the string, then it provides the index of the letter. Otherwise, say that letter does not exist in the string.