# Functions and Predicates Relating to Unit strings

        `ustring? :  string --> boolean`

Purpose: tests for unit string


        `digit? : ustring --> boolean`

Purpose: tests if a string is in ["0", "1", …, "9"]


        `uppercase? : ustring --> boolean`

Purpose: tests if a string is in ["A", "B", …, "Z"]


        `lowercase? : ustring --> boolean`

Purpose: tests if a string is in ["a", "b", …, "z"]


        `letter? : ustring --> boolean`

Purpose: tests if a string is lower-case or upper-case


        `whitespace? : ustring --> boolean`

Purpose: tests if a string is in  ["c#9;" "c#10;" "c#11;" "c#12;" "c#13;" "]


        `ustring-upcase : ustring --> ustring`

Purpose: converts a lower-case letter to upper case


        `ustring-downcase : ustring --> ustring`

Purpose: converts an upper-case letter to lower case


# String Predicates

        `ustring?  : string --> boolean`


        `digit-string? : string --> boolean`

Purpose: tests if a string represents an unsigned integer


        `string-prefix? : string --> string --> boolean`

Purpose: tests if a string is a prefix of another string


        `string-suffix? : string --> string --> boolean`

Purpose: tests if a string is a suffix of another string


        `substring? : string --> string --> boolean`

Purpose: tests if a string is a substring of another string


        `string-every? : (ustring --> boolean) --> string --> boolean`

Purpose: tests if a predicate of unit strings applies to all unit strings of a string


        `string-any? : (ustring --> boolean) --> string --> boolean`

Purpose: tests if a predicate of unit strings applies to at least one unit string of a string

## Length Functions

```
string-length : string --> integer
```
Purpose: returns the number of unit strings in a string
```
string-prefix-length : string --> string --> integer
```
Purpose: returns the length of the longest prefix common to two strings

```
string-suffix-length : string --> string --> integer
```
Purpose: returns the length of the longest suffix common to two strings

## Case Conversion

```
ustring-upcase : ustring --> ustring
ustring-downcase : ustring --> ustring
```

```
string-upcase : string --> string
```
Purpose: converts all lower-case letters of a string to upper case

```
string-downcase : string --> string
```
Purpose: converts all upper-case letters of a string to lower case

## Comparison

```
=     : A --> A --> boolean
!=    : A --> A --> boolean
<=str : string --> string --> boolean
>=str : string --> string --> boolean
<str  : string --> string --> boolean
>str  : string --> string --> boolean
```

## Selection

```
string-ref : integer --> string --> ustring
```
Purpose: returns the unit string at a specified position of a given string (replaces **pos**)

```
string-take : integer --> string --> string
```
Purpose: returns the prefix of a specified length of a given string

```
string-drop : integer --> string --> string
```
Purpose: returns a string without the prefix of a specified length

```
string-split : integer --> string --> (string * string)
```
Purpose: returns the pair consisting of the prefix of length N of S and the remaining suffix.

```
string-take-right : integer --> string --> string
```
Purpose: returns the suffix of a specified length of a given string

```
string-drop-right : integer --> string --> string
```
Purpose: returns a string without the suffix of a specified length

```
substring : integer --> integer --> string --> string
```
Purpose: returns a substring (bounds specified) of a string

```
string-trim-left : (ustring --> boolean) --> (string --> string)
```
Purpose: returns the string obtained from a string S by dropping the longest prefix of S all of whose unit strings
    satisfy a given predicate

2

```
string-trim-right : (ustring --> boolean) --> (string --> string)
```
Purpose: returns the string obtained from a string S by dropping the longest suffix of S all of whose unit strings satisfy a given predicate

```
string-trim : (ustring --> boolean) --> (string --> string)
```
Purpose: trims a string both on the left and the right

```
string-pad : ustring --> integer --> string --> string
```
Purpose: returns the string of length N obtained from string by padding it to length N with copies of a given unit string

## Searching and Counting
```
string-index : string --> string --> integer
```
Purpose: returns the start index of a substring of a given string, or -1

```
string-index-last : string --> string --> integer
```
Purpose:  the start index of the last occurrence of a substring, or -1.

```
string-count : string --> string --> integer
```
Purpose: counts the number of occurrences of a string as a substring of given string

```
string-count-ustrings : (ustring --> boolean) --> string --> integer
```

## Replace and Delete
```
string-replace-all : string --> string --> string --> string
```
Purpose: replaces all occurrences of a substring of a string by some other string

```
string-delete-all : string --> string --> string
```
Purpose: deletes all occurrences of a substring of a string

```
delete-substring : integer --> integer --> string --> string
```
Purpose: deletes a substring

## Tokenising and Joining
```
string-tokenise : string --> (ustring --> boolean) --> (list string)
```
Purpose: produces a list of tokens for a given string and separators defined by a predicate of unit strings

```
string-join : string --> (list string) --> string
```
Purpose: returns the string obtained by inserting a string between every two strings in a list of strings

```
string-interpose : string --> string --> string
```
Purpose: returns the string obtained by inserting a string between every two unit strings of a string

## Strings and Lists
```
string->list : string --> (list ustring)
```
Purpose: returns a list of the unit strings of a string

```
list->string : (list string) --> string
```
Purpose: returns the string obtained by concatenating the strings in a given list

```
string-tokenise : string --> (ustring --> boolean) --> (list string)
string-join : string --> (list string) --> string
```

## Higher Order Functions

```
string-every? :  (ustring --> boolean) --> string --> boolean
string-any? :  (ustring --> boolean) --> string --> Boolean

string-map : (ustring --> string) --> string --> string
```
Purpose: creates a string by applying a function to all the constituent unit strings of a string

```
string-filter : (ustring --> boolean) --> string --> string
```
Purpose: returns the string of all unit strings of a string for which a given predicate is true

```
string-reduce : (ustring --> A --> A) --> A --> string --> A
```
Purpose: returns the reduction of a string with respect to a given function

```
string-count-ustrings : (ustring --> boolean) --> string --> integer
```
Purpose: counts the number of unit strings of a string which satisfy a given predicate

```
string-trim-left : (ustring --> boolean) --> (string --> string)
string-trim-right : (ustring --> boolean) --> (string --> string)
string-trim : (ustring --> boolean) --> (string --> string)
```

## Miscellaneous

```
string-reverse : string --> string
```
Purpose: reverses a string

```
string-n-copy : integer --> string --> string
```
Purpose: returns a string of n copies of a string

## String to Number Conversion

```
string->number : string --> number
```
Purpose: converts a string to a Shen number

## Radix Conversion

```
decimal->radixB : integer --> integer --> string
```
Purpose: converts from decimal to radix B

```
radixB->decimal : string --> integer --> integer
```
Purpose: converts from radix B to decimal

```
radixB->radixC  : string --> integer --> integer --> string
```
Purpose: converts from radix B to radix C

```
string->integer : macro - no signature
```
Purpose: see notes