



Assignment 3

CSI2120 Programming Paradigms

Winter 2019

Due on April 5th before 11:00 pm in Virtual Campus

6 marks

There are [10 points] in this assignment. The assignment is worth 6% of your final mark.

All code must be submitted in a scm file. Screenshots, files in a format of a word editor, pdfs, handwritten solutions, etc. will not be marked and receive an automatic 0.

Question 1. [1 point]

Use the built-in function map to replace every number by its reciprocal value in the list. Define a global 1over and use 0 as the reciprocal value for 0, i.e., $0 = 1/0$.

```
(1over '(0 2 3 4 12 0 0 1 0))  
⇒ '(0 1/2 1/3 1/4 1/12 0 0 1 0))
```

Question 2. [2 points]

Implement Newton-Rhapson's method for root finding of function in one dimension. Newton-Rhapson is defined by the iteration

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

where x_n is the current estimate of the root, $f(x_n)$ is the function evaluated at x_n and $f'(x_n)$ is the derivative of the function evaluated at x_n . Your global definition of the function must take three arguments, the current estimate of the root, the function and the derivative of the function, i.e.,

```
; (newtonRhap x f fx)
```

Your routine must stop the iterations if the change in the solution is less than a tolerance. For this purpose use a global define, i.e.,

```
(define TOL 1e-6)
```

See next page for examples.

Examples:

```
(newtonRhap 0.1 sin cos)
⇒ 0
(newtonRhap 2.0 (lambda (x) (- (* x x) x 6))
               (lambda (x) (- (* 2 x) 1)))
⇒ 3.0
(newtonRhap -20.0 (lambda (x) (- (* x x) x 6))
                 (lambda (x) (- (* 2 x) 1)))
⇒ -2.0000000000000118
```

Question 3. [3 points]

Implement a routine `p_cos` which calculates the cosine of an angle in radians. Use the following product approximation of cosine

$$\cos(x) = \prod_{n=1}^{\infty} \left[1 - \frac{4x^2}{\pi^2(2n-1)^2} \right]$$

Use as many terms until the change with the next term is less than a tolerance. For this purpose use again the global define,

```
(define TOL 1e-6)
```

Examples:

```
(p_cos 0)
⇒ 1
(p_cos (/ pi 2))
⇒ 0.0
```

You are allowed extra global helper functions.

Question 4. [5 points]

You are not allowed to use any of the built-in string processing function for this question. You can assume that all lists only contain characters.

- a) Write a predicate `separator?` that returns true if a character is a space, tab or newline and false otherwise. In Scheme these characters are written `#\space`, `#\tab` and `#\newline` respectively. The built-in predicate `char=?` compares two characters for equality.

Example:

```
(separator? #\space)
```

```
⇒ #t
```

```
(separator? #\b)
```

```
⇒ #f
```

- b) Write a function `cpy` that copies all characters from an input list into an output list until a separator is encountered.

```
(cpy ' (#\H #\e #\l #\l #\o #\space #\W #\o #\r #\l #\d))
```

```
⇒ ' (#\H #\e #\l #\l #\o)
```

- c) Write a function `drop` that removes all characters from an input list until a separator is encountered and returns the remaining list.

```
(drop ' (#\H #\e #\l #\l #\o #\newline #\W #\o #\r #\l #\d))
```

```
⇒ ' (#\W #\o #\r #\l #\d)
```

- d) Write a predicate `same?` that compares two list of characters and return true if the characters in the first input list up to a separator are the same as in the second list.

```
(same? ' (#\H #\e #\l #\l #\o #\tab #\W #\o #\r #\l #\d)
```

```
      ' (#\H #\e #\l #\l #\o))
```

```
⇒ #t
```

```
(same? ' (#\H #\e #\l #\l #\o #\space #\W #\o #\r #\l #\d)
```

```
      ' (#\W #\o #\r #\l #\d))
```

```
⇒ #f
```

continued on next page.

- e) Write a function `replace` that replaces a list of characters in the input list between separators with another set of characters.

```
(replace
'(#\a #\space #\b #\i #\r #\d #\space #\e #\a #\t #\s #\space
#\a #\space #\t #\o #\m #\a #\t #\o)
'(#\a)
'(#\t #\h #\e))
⇒ ' (#\t #\h #\e #\space #\b #\i #\r #\d #\space #\e #\a #\t #\s
   #\space #\t #\h #\e #\space #\t #\o #\m #\a #\t #\o)
```

Or more readable:

```
(list->string (replace (string->list "a bird eats a tomato")
(string->list "a") (string->list "the")))
⇒ "the bird eats the tomato"
```