SAINT MARY'S UNIVERSITY SINCE 1802
One University. One World. Yours.

Department of Mathematics and Computing Science
CSCI 3430.1 - Principles of Programming Languages

Assignment #3

Assigned : 22 January 2017
Due : 31 January 2017 (11:55PM)

Hand in source code only. Please add all source code into one big zip file and post that to Moodle, it won't take more than **ONE** file. If it's too large to upload to Moodle, you likely have more than source code. If you are still stuck and can't upload the file, email it to me, cc to yourself and see me in class.

LISP "programs" are simply functions that return values when called interactively from the interpreter with appropriate input. There is no real input or output in the usual way. The program simply returns the answer. Please verify your answers on the **CLISP** interpreter on cs.smu.ca. If you don't use that one, PLEASE let me know which LISP variant you are using and what it is running on, but be warned that I'll be marking it using CLISP. Also please note that LISP programs do not (and must not in these assignments) make use of variables (let or set or setf or any other form of variable assignment), or other iterative structures (for, do, while, repeat, loop). Functions using such structures will be marked zero. You MUST use recursion!

1. (1 pt) What happens when you execute the following code in a COMMON LISP interpreter? Why?

```
(defun should-be-constant ()
  '(one two three))

(let ((stuff (should-be-constant)))
  (setf (third stuff) 'bizarre))

(should-be-constant)
```

2. (1 pts) Write the LISP recursive factorial program. It's sort of the math version of "hello world" so you can find code online to answer to this. Note you won't have to test it with numbers below 0 or greater than 10 so input traps are not required. However it will handle large numbers than C or Java (for a good time try (fact 1000) or (fact 12345))

```
* (fact 0)
1
* (fact 1)
1
* (fact 4)
24
* (fact 10)
3628800
```

3. (2 pts) Write code that computers N Fibonacci numbers (where N is the function parameter) . Finonacci numbers are the numbers where every number after the first two is the sum of the two preceding ones

```
* (fib 0)
()
* (fib 2)
( 1 1 )
* (fib 6)
( 1 1 2 3 5 8 )
```

4. (2 pts) Write a function called "notnums" that will take a list of arbitrary length and nesting complexity and return the number of non-numbers the list contains, anywhere (i.e., at any level of nesting) in the list. Your function should be able to handle the following input:

```
* (notnums '(a 9 (5 b) c (d (e 7))))
5
* (notnums '(7 a 8 b c 9 10))
3
* (notnums '(1 (2 (3)) y 5 (x) 9))
2
* (notnums '(a 1 (b 2) ())
2
* (notnums '(* b (a (- 2) c 3) (+ 5))
6
```

5. (2 pts) Write a function "isprime" that will take as input one number and test if it is a prime number. The answer should return a T or NIL and work for number greater than one and include the special case for the even number 2. Also show the assembler language version of the function.

```
* (isprime 10)
NIL
* (isprime 11)
T
* (isprime 123456)
NIL
* (isprime 2)
T
```

6. (2 pts) Write a function "hfxnum" that validates phone numbers for the Halifax area and returns either T or NIL. A phone number takes the form of (3 digit area code)-(3 digit local exchange)-(4 digit subscriber phone number). Note there are actually two area codes in use in the Halifax area 782 and 902, each with a subset of exchange numbers:

(782) = 486 510 610 845 and

(902) = 209 210 219 220 221 222 223 225 229 233 237 240 244 252 266 268 292 293 329 332 333 334 337 342 344 346 347 359 377 384 399 401 402 403 404 405 406 407 412 414 415 420 421 422 423 424 425 426 427 428 429 430 431 440 441 442 443 444 445 446 448 449 450 451 452 453 454 455 456 457 458 459 470 471 473 474 475 476 477 478 479 480 482 483 484 486 487 488 489 490 491 492 493 494 495 496 497 498 499 501 503 520 536 551 558 559 568 571 576 579 580 593 700 701 702 703 704 705 706 707 708 714 717 718 719 720 721 722 772 789 797 800 801 802 809 817 818 827 830 868 873 876 877 878 880 889 891 949 981 982 989 997 998 999.

More or less numbers should return a NIL. Any character should return a NIL. Anything except the above area codes and exchanges should return a NIL.

```
* (hfxnum '(1 2))
NIL
* (hfxnum '(9 0 2 4 4 9 1 2 3 4))
T
```

```
* (hfxnum ‘(9 0 2 7 4 2 1 2 3 4))
NIL
* (hfxnum ‘(7 8 2 4 8 6 1 2 3 4))
NIL
* (hfxnum ‘(6 1 3 4 4 9 1 2 3 4))
NIL
```

PS: There’s a many CLISP tutorials online like:	http://cs.gmu.edu/~sean/lisp/LispTutorial.html
A great but ancient online reference book is:	http://www.cs.cmu.edu/Groups/AI/html/cltl/clm/clm.html
And there’s always the online CLISP user manual:	http://www.clisp.org/