Assignment 4
CMSPC 461: Programming Language Concepts

Prof. Suman Saha, Spring 2023

Due: 11:59 PM, $17^{th} March, 2023$

Instructions:
For this assignment, you need to submit your solution as one **single file named as "code.rkt" to Gradescope.** If your file name is not "code.rkt" or you define a function name that is different from what is specified in the question, **Gradescope will not grade your assignment.** Please clearly mark your answers using comments so that we can tell the correspondence between your code and questions. You may NOT use any of Schemes imperative features (assignment/loops) or anything else not covered in class. You should use Racket (https://racket-lang.org) for your implementation. For all problems, you can assume all inputs obey the types as specified in a problem. You will have access to some test cases when you submit your code to Gradescope.
(Kindly refer syllabus for late submission and academic integration policies.) **We will check your code for plagiarism

---

Question 1 (20 pt)
You have opened a credit card company with great cashback benefits for college students. Your company provides yearly cashback to the customer based on how much they spend in a year. The cashback works as follows:

- 1% for the first $500 spent
- 1.5% for next $500 (i.e. between $500-$1000)
- 2% for next $1000 (i.e. between $1000-$2000)
- 3% for everything above $2000

Example: Let us say a student spent $100 using this particular credit card in a year, then the student will get $1 as cashback. Another student who spent $2500 will get $47.5

Implement the function cashBack that takes year_spent as a parameter and computes the corresponding cashback amount.

Question 2 (20 pt)

Define a recursive function merge, which takes two sorted lists of numbers, and returns one merged list where all numbers are sorted in decreasing order. Assume that all elements are sorted in decreasing order. For example:

```
(merge '( 9 8) '( ) ) ; returns (9 8)
(merge '( 7 5 2) '( 9 6 2 ) ) ; returns (9 7 6 5 2 2)
(merge '( 2 2 2) '( 3 2 ) ) ; returns (3 2 2 2 2)
```

Question 3 (20 points)

Implement the function myGCD that recursively used Euclid's algorithm to calculate the greatest common divisor of two numbers. Following is the algorithm:

$$\text{myGCD(x,y)=}\begin{cases} x & \text{if } y = 0 \\ myGCD(y, remainder(x, y)) & \text{if } y > 0 \end{cases}$$

Scheme has a built-in remainder function, which you can use for this problem. Also, **Scheme already has a built-in gcd function, *which you shouldn't use*.**

Question 4 (20 points)

Write a higher-order function *ncall* that takes three parameters: *n, f, x*; it returns x when $n = 0$, returns $f(x)$ when $n = 1$, returns $f(f(x))$ when $n = 2$, etc.
That is, it returns the result of calling f on x, for n times.
For instance, invoking $ncall$ with $n = 4$, a function that adds 1, and $x = 2$ should return 6.
Another example in scheme: a call to `(ncall 4 (lambda (x) (+ x 2)) 2)` should return 10. In this case f(x)=x+2 (the lambda function) thus for x=2 we are calling f 4 times on x i.e.
f(f(f(f(x))))=f(f(f(f(2))))=f(f(f(4)))=f(f(6)))=f(8)=10

Question 5 (20 points)

Implement the following pseudo code $myFunc$ that takes two non-negative parameters in scheme:

```
begin myFunc(x,y):
    if x==0:
        return y+1
    else if x>0 and y=0:
        return myFunc(x-1,1)
    else if x>0 and y>0:
        return myFunc(x-1,myFunc(x,y-1))
```

Note: This is a computationally intensive function. We will not evaluate this using larger numbers (i.e $x > 4$).
Example: (myFunc 3 3) should return 61