CS-GY 6373 Programming Languages Midterm Fall 2020

Last Name:_____ First Name:_____NetID:_____

1) (20 pts) For the following forms apply β-reduction and α-conversion where appropriate. Explain each step.
   a) (λx.x)(λx.x)
   a) (λx.x x) (λx. λy.x x)
   b) ((λx.(x y))(λz.z))
   c) (λz.z) (λy.y y) (λx.x a)
   d) (λz.z) (λz.z z) (λz.z y)
   e) (λx.λy.x y y) (λa.a) b
   f) (λx.x x) (λy.y x) z
   g) (λx. (λy. (x y)) y) z
   h) ((λx.x x) (λy.y)) (λy.y)
   i) (((λx. λy.(x y))(λy.y)) w)

2) (20 pts) Consider the following BNF grammar for a language with two infix operators @ and $ and terminals a b c. To receive credit, you must attach a detailed explanation for each of your answers that show the rationale for your answer.

   S ::= A | S @ A

   V ::= S | S $ V

   A ::= a | b | c

   a) What is the associativity of the @ operator: (a) left (b) right (c) neither
   b) What is the associativity of the $ operator: (a) left (b) right (c) neither
   c) Which operator has higher precedence: (a) @ (b) $ (c) neither
   d) The grammar is: (a) left recursive (b) right recursive (c) both left and right recursive (d) neither left nor right recursive (c)
   e) Draw a parse tree for the following string: a @ b @ c $ a $ b

3) (15 pts) Write programs in Haskell, Python, and Scheme that reverses a simple list of integers using recursion; specifically, in tail recursive form. Do not use higher order functions such as “reverse”, comprehensive lists or functions from a library.

4) (5 pts) Prove that the following grammar is ambiguous
   <S> → <A>
   <A> → <A> + <A> | <id>
   <id> → a | b | c

Last Name:_____ First Name:_____NetID:_____

5) Given the following grammar:

<assign> → <id> = <expr>

<id> → A | B | C

<expr> → <expr> + <term>
| <term>

<term> → <term> * <factor>
| <factor>

<factor> → (<expr>)
| <id>

a) (10 pts) Rewrite the BNF to give + precedence over * and force + to be right associative.
b) (10 pts) Rewrite the BNF to add the ++ and -- unary operators

6) (20 pts) In your own words what are the key elements of a Functional Programming Language. Compare this with an Imperative Programming Language. What operations are available in Programming Language and how do they differ from each other.