# Programming Languages and Compilers
## Final Exam

*This is an open exam, meaning that you are free to use books, notes, computers, and the internet. However, any form of collaboration is prohibited. Complete the exam, and email it as a PLAIN TEXT under the subject "CISC 7120 Final" to nzhou@brooklyn.cuny.edu by midnight on Wednesday, December 15.*

### Question 1 (15 points)

Give a regular expression for each of the following languages over $\Sigma = \{0, 1, 2\}$.

1. All strings that begin with 1 and end with 2.
2. All strings that contain exactly three 1's. For example, `"0101012"` is valid.
3. All three-digit decimal integers that are divisible by 3. For example, `"0"`, `"102"` are valid, but `"100"` is not.

## Question 2 (15 points)

The following DFAs define three languages over $\Sigma = \{0, 1\}$. Describe each of the languages and give a regular expression for it.

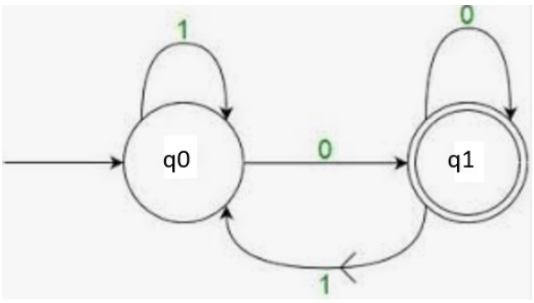


Figure 1: (DFA-1)

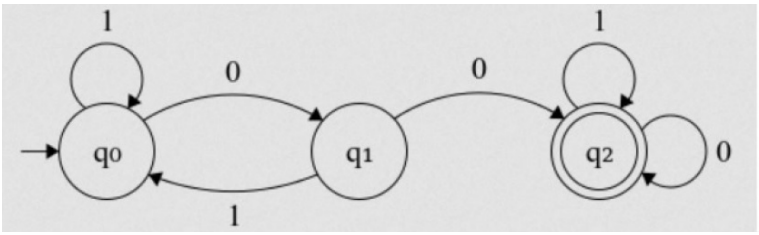


Figure 2: (DFA-2)

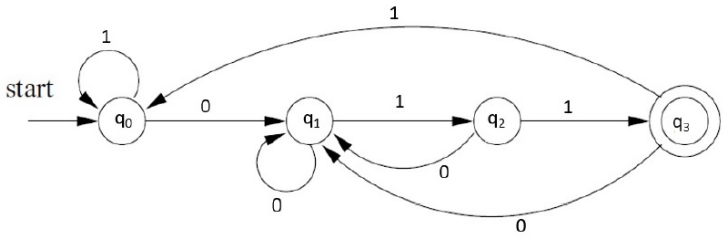


Figure 3: (DFA-3)

## Question 3 (20 points)

Consider the following CFG:

```
E ::= (E + E) | (E * E) | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

1. Is the grammar ambiguous? Why?
2. Give a left-most derivation for the expression `"(2 * ( 3 + 4) )"`.
3. Write a function in a programming language of your choice that takes an expression as a string and returns the evaluated value. For example, for the above expression, it returns `14`. If the string is not a valid expression, the function throws an exception.

## Question 4 (20 points)

Write each of the following functions in Picat or Haskell *using recursion*. No higher-order functions or list comprehensions can be used in the implementations.

1. `member(x,lst)`: This function checks if `x` occurs in `lst`.
2. `sorted_list(lst)`: This function checks if `lst` is sorted in ascending order.
3. `mode(lst)`: This function returns the most frequently occurring element in `lst`. For example, the returned value for `lst = [1,3,1,3,2,1]` is 1. If there are more than one most frequent element, then the function can return any one of them.

## Question 5 (10 points)

Consider the function `gen` defined below in Picat and Haskell.

```
%% In Picat
gen(N) = gen(N,[],0,0).

gen(0,Str,Na,Nb) = [Str], Na > Nb => true.
gen(0,Str,Na,Nb) = [].
gen(N,Str,Na,Nb) = Res =>
    Res1 = gen(N-1,[a|Str],Na+1,Nb),
    Res2 = gen(N-1,[b|Str],Na,Nb+1),
    Res = Res1 ++ Res2.

-- Haskell
gen n  = gen_aux n [] 0 0

gen_aux 0 str na nb
    | na > nb = [str]
    | otherwise = []
gen_aux n str na nb = res1 ++ res2
    where res1 = gen_aux (n-1) ('a':str) (na+1) nb
          res2 = gen_aux (n-1) ('b':str) na (nb+1)
```

1. What is the output of the function call `gen(4)`?
2. What is the return value for a given value n, in general?
3. (Extra 5 points) The function definition is not tail recursive. Convert it into tail recursion.

## Question 6 (20 points)

Design a data structure for binary trees and write each of the following functions on binary trees in Haskell, Java, C++, Picat, or Python:

1. `member(x,btree)`: This function checks if `x` occurs in `btree`. The binary tree `btree` is not necessarily a binary search tree.

2. `one_child_node_values(btree)`: This function takes a binary tree and returns a list of values of the nodes in the tree that have exactly one child. The order of the values in the list is not important.

3. `shallowest_leaf(btree)` (extra 5 points): This function returns the values in a shallowest leaf node in `btree`. If there are multiple such leaves, then the function returns the left-most one.