

Problem Set # 2

The grammar below is a simplification of C++ (global) function headers including pointer notation, e.g.:

```
int * my_function( int * A, int n )
```

In the following grammar, words beginning with capital letters are variables. Words all in lower case are terminals (tokens). Special characters including as “(”, “)”, “,”, and “*” are terminals.

```
Fun -> Type identifier ( Plist )  
Type -> int Slist | double Slist  
Slist -> * Slist | epsilon  
Plist -> Type identifier MoreList  
MoreList -> , Type identifier MoreList | epsilon
```

The FIRST and FOLLOW sets for this grammar are as follows:

First sets:

```
FIRST(Fun) = { int double }  
FIRST(Type) = { int double }  
FIRST(Slist) = { * epsilon }  
FIRST(Plist) = { int double }  
FIRST(MoreList) = { epsilon , }
```

Follow sets:

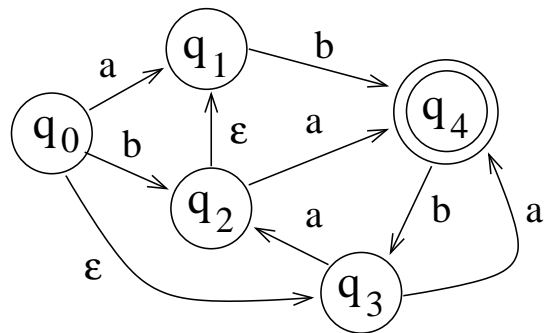
```
FOLLOW(Fun) = { eof }  
FOLLOW(Type) = { identifier }  
FOLLOW(Slist) = { identifier }  
FOLLOW(Plist) = { ) }  
FOLLOW(MoreList) = { ) }
```

Your task:

1. Give a predictive parsing table for the given grammar.

DFA, NFA, and Regular Expressions

1. Given the following NFA, construct an equivalent DFA.



2. Given the following regular expression, convert to an equivalent NFA.

$$(a | b)^* (b | c)$$