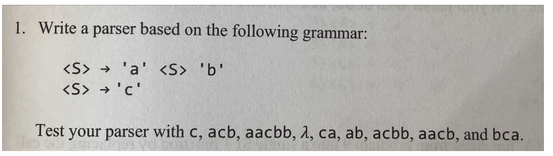
Lab 4

Saturday, September 17, 2022 3:57 PM

Chapter 4, problems 1, 2, 3



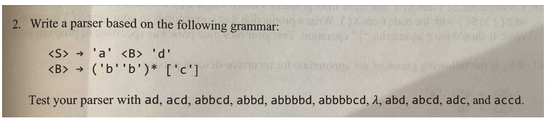
1. Write a parser based on the following grammar:

```
<S> → 'a' <S> 'b'
<S> → 'c'
```

Test your parser with c, acb, aacbb, λ, ca, ab, acbb, aacb, and bca.

```
def S():
    if token == 'a':
        advance()
        S()
        consume('b')
    elif token == 'c':
        advance()
    else:
        raise RuntimeError('expecting a or c')
```

```
beesannekurzum@Beesannes-MacBook-Pro Lab4 % python3 q1.py c
beesannekurzum@Beesannes-MacBook-Pro Lab4 % python3 q1.py acb
beesannekurzum@Beesannes-MacBook-Pro Lab4 % python3 q1.py aacbb
beesannekurzum@Beesannes-MacBook-Pro Lab4 % python3 q1.py ''
expecting a or c
beesannekurzum@Beesannes-MacBook-Pro Lab4 % python3 q1.py ca
Garbage following <S>-string
beesannekurzum@Beesannes-MacBook-Pro Lab4 % python3 q1.py ab
expecting a or c
beesannekurzum@Beesannes-MacBook-Pro Lab4 % python3 q1.py acbb
Garbage following <S>-string
beesannekurzum@Beesannes-MacBook-Pro Lab4 % python3 q1.py aacb
Expecting b
beesannekurzum@Beesannes-MacBook-Pro Lab4 % python3 q1.py bca
expecting a or c
```



2. Write a parser based on the following grammar:

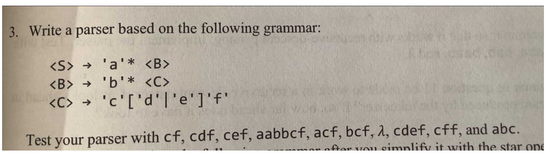
```
<S> → 'a' <B> 'd'
<B> → ('b'*'b')* ['c']
```

Test your parser with ad, acd, abbcd, abbd, abbbbd, abbbbcd, λ, abd, abcd, adc, and accd.

```
def S():
    if token == 'a':
        advance()
        B()
        consume('d')
    elif token == 'd':
        advance()
    else:
        raise RuntimeError ('expecting a')

def B():
    while token == 'b':
        advance()
        consume('b')
    if token == 'c':
        advance()
```

```
beesannekurzum@Beesannes-MacBook-Pro Lab4 % python3 q1.py ad
beesannekurzum@Beesannes-MacBook-Pro Lab4 % python3 q1.py acd
beesannekurzum@Beesannes-MacBook-Pro Lab4 % python3 q1.py abbcd
beesannekurzum@Beesannes-MacBook-Pro Lab4 % python3 q1.py abbd
beesannekurzum@Beesannes-MacBook-Pro Lab4 % python3 q1.py abbbbd
beesannekurzum@Beesannes-MacBook-Pro Lab4 % python3 q1.py abbbbcd
beesannekurzum@Beesannes-MacBook-Pro Lab4 % python3 q1.py ''
expecting a
beesannekurzum@Beesannes-MacBook-Pro Lab4 % python3 q1.py abd
Expecting b
beesannekurzum@Beesannes-MacBook-Pro Lab4 % python3 q1.py abcd
Expecting b
beesannekurzum@Beesannes-MacBook-Pro Lab4 % python3 q1.py adc
Garbage following <S>-string
beesannekurzum@Beesannes-MacBook-Pro Lab4 % python3 q1.py accd
Expecting d
```



3. Write a parser based on the following grammar:

```
<S> → 'a'* <B>
<B> → 'b'* <C>
<C> → 'c'['d'|'e']'f'
```

Test your parser with cf, cdf, cef, aabbcf, acf, bcf, λ, cdef, cff, and abc.

```
def S():
    while token =='a':
        advance()
    B()

def B():
    while token =='b':
        advance()
    C()

def C():
    consume('c')
    if token == 'd':
        advance()
    elif token == 'e':
        advance()
    consume('f')
```

```
beesannekurzum@Beesannes-MacBook-Pro Lab4 % python3 q1.py cf
beesannekurzum@Beesannes-MacBook-Pro Lab4 % python3 q1.py cdf
beesannekurzum@Beesannes-MacBook-Pro Lab4 % python3 q1.py cef
beesannekurzum@Beesannes-MacBook-Pro Lab4 % python3 q1.py aabbcf
beesannekurzum@Beesannes-MacBook-Pro Lab4 % python3 q1.py acf
beesannekurzum@Beesannes-MacBook-Pro Lab4 % python3 q1.py bcf
beesannekurzum@Beesannes-MacBook-Pro Lab4 % python3 q1.py ''
Expecting c
beesannekurzum@Beesannes-MacBook-Pro Lab4 % python3 q1.py cdef
Expecting f
beesannekurzum@Beesannes-MacBook-Pro Lab4 % python3 q1.py cff
Garbage following <S>-string
beesannekurzum@Beesannes-MacBook-Pro Lab4 % python3 q1.py abc
Expecting f
```