ECE 553: Compilers
Midterm

Name: NetID:

There are 9 questions, with the point values as shown below. You have 75 minutes with a total of 75 points. Pace yourself accordingly.

This exam must be individual work. You may not collaborate with your fellow students. However, this exam is open book and open notes. You may use any printed materials, but no electronic nor interactice resources.

I certify that the work shown on this exam is my own work, and that I have neither given nor received improper assistance of any form in the completion of this work.

Signature:

#	Question	Points Earned	Points Possible
1	SML		5
2	Regular Languages		5
3	NFA to Regexp		10
4	NFA to DFA		10
5	Regexp to NFA		10
6	Context Free Languages		5
7	LL Parsing		10
8	LR Parsing		10
9	Types		10
	Total		75
	Percent		100

Question 1 SML [5 pts]

Part A: Evaluate the following SML expressions (1 point each):

1.
```
let val a = 42
    val b = 20
in
  (if (a > b) then a else b) * 2
end
```

2.
```
let val a = 7
    val q = 6
    val b = let val a = 9
                val q = 15
            in
              a + q
            end
in
  a - b  + q
end
```

3.
```
let fun f (a, 0) = a
      | f (a, b) = f (a*2+b, b-1)
in
    f (4, 2)
end
```

Part B: Write down the **type** for each of these SML functions (1 point each):

1. `fun f  x y  = (x y) ^ (x 3)`

2.
```
fun f(x,[]) = []
    | f(x,a::l) = case x a of
                    SOME y => y::f(x,l)
                  | NONE => f(x,l)
```

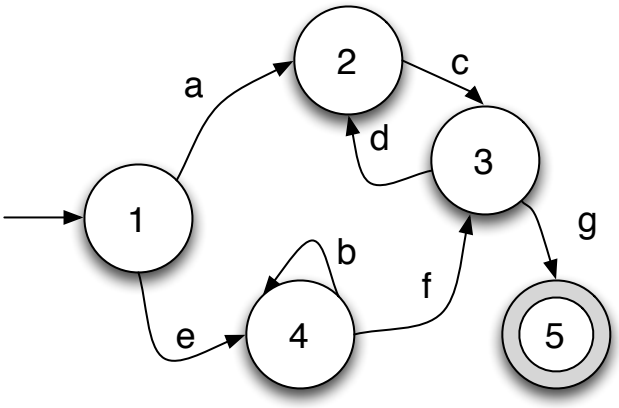
Question 2 Regular Languages [5 pts]

1. Write a regular expression for all strings of `0`s and `1`s which does not contain the substring `0011`.

2. Write a regular expression for all strings of `x`s and `y`s where the number of `y`s between any two `x`s is a multiple of 3.

Question 3 NFA to Regexp [10 pts]

Convert the following NFA to a regular expression :

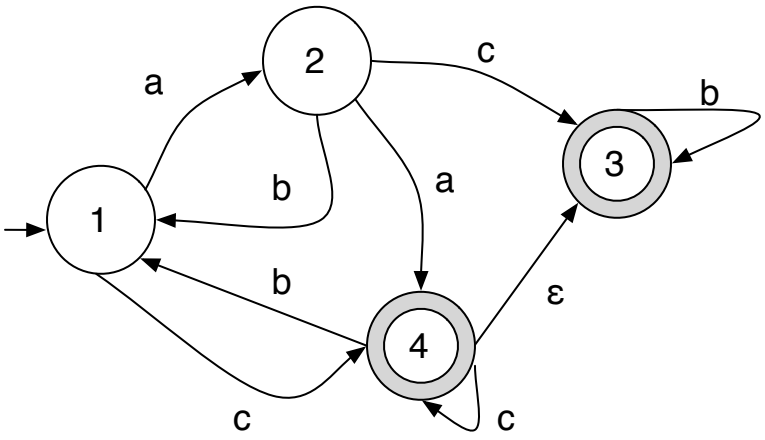


Workspace for question 3

Workspace for question 3

Question 4 NFA to DFA [10 pts]

Convert the following NFA to a DFA:



Question 5 Regexp to NFA [10 pts]

Draw an NFA for the following regular expressions:

1. (a |c) (b | d)

2. a*(bc)*

3. ((a(b|c)*d) | e*)f

Question 6 Context Free Languages [5 pts]

Write context free grammars for the following languages (your grammar does not have to be LR(1), LL(1) etc):

1. The language described by the regular expression ((a(b|c)*d) | e*)f

2. The language of sequences of `0`s and `1`s that are palindromes (the same forwards as backwards):

Question 7 LL Parsing [10 pts]

Consider the following grammar:

```
S -> A x B
   | C
A -> A x z A
   | y
B -> b B b
   |
C -> C y
   | B
```

- Which non-terminals (if any) can derive empty? (1 point)

- What are the FIRST sets of Q and S? (1 point)

  .

- What are the FOLLOW sets of Q and S? (1 point)

- This grammar can not be parsed by an LL(0) or LL(1) parser. Explain why not (2 points).

- Rewrite the grammar so that it accepts the same langauge, but can be parsed by an LL(1) parser (5 points).

Question 8 LR Parsing [10 pts]

Consider the following grammar:

```
0: S -> X $
1: X -> X a X b
2: X -> c
```

Below is the state table (`.grm.desc` file) from `ml-yacc`, with the non-error actions removed (replaced by blanks). Your job is to fill in the missing actions (`shift N`, `goto N`, `reduce by rule R` where N is the appropriate state number, and R is the appropriate rule number). The first (State 0) and last (State 6) are done for you:

```
state 0:

        S : . X

        c       shift 2

        S       goto 6
        X       goto 1

        .       error


state 1:

        S : X .
        X : X . a X b

        a       ________________________


        .       ________________________


state 2:

        X : c .




        .       ________________________


state 3:

        X : X a . X b

        c       ________________________

        X       ________________________
```

```
        .       ------------------------


state 4:

        X : X . a X b
        X : X a X . b

        a       ------------------------
        b       ------------------------


        .       ------------------------


state 5:

        X : X a X b .



        .       ------------------------


state 6:
        EOF     accept
        .       error
```

Question 9 Types [10 pts]

1. Show the typing derivation for the Tiger statement `y := if f(r.a) then 3 else y + 2`. You may assume that your initial environment (Γ_0) has the following mappings (in addition to the base Tiger environment):
   Γ_0(y) = int
   Γ_0(r) = Record(a:int, b:string)
   Γ_0(f) = int $\rightarrow$ int

2. If `bat` $\sqsubseteq$ `animal` and `fruitbat` $\sqsubseteq$ `bat`, then which of the following types are a subtype of `bat` $\rightarrow$ `bat` (circle *all* types that are $\sqsubseteq$ `bat` $\rightarrow$ `bat`)?

animal $\rightarrow$ animal	animal $\rightarrow$ bat	animal $\rightarrow$ fruitbat
bat $\rightarrow$ animal	bat $\rightarrow$ bat	bat $\rightarrow$ fruitbat
fruitbat $\rightarrow$ animal	fruitbat $\rightarrow$ bat	fruitbat $\rightarrow$ fruitbat

3. Given the subtyping relationship in the previous part, which of the following types are subtypes of

$$(\texttt{bat} \to \texttt{bat}) \to \texttt{bat}$$

Again, circle *all* that apply:

(fruitbat→fruitbat)→ fruitbat	(fruitbat→fruitbat)→ bat	(fruitbat→fruitbat)→ animal
(fruitbat→ bat) → fruitbat	(fruitbat→ bat) → bat	(fruitbat→ bat) → animal
(fruitbat→ animal) → fruitbat	(fruitbat→ animal) → bat	(fruitbat→ animal) → animal
(bat→fruitbat)→ fruitbat	(bat→fruitbat)→ bat	(bat→fruitbat)→ animal
(bat→ bat) → fruitbat	(bat→ bat) → bat	(bat→ bat) → animal
(bat→ animal) → fruitbat	(bat→ animal) → bat	(bat→ animal) → animal
(animal→fruitbat)→ fruitbat	(animal→fruitbat)→ bat	(animal→fruitbat)→ animal
(animal→ bat) → fruitbat	(animal→ bat) → bat	(animal→ bat) → animal
(animal→ animal) → fruitbat	(animal→ animal) → bat	(animal→ animal) → animal