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 notes, so you may use your class notes.

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 = 12
    val b = 20
in
    (if b - a > 7
     then 2
     else 3)::[a,b*2]
end
```

2.
```
let fun f a = let val temp = 3 + a
              in
               fn b => temp * b
              end
    val g = f 4
in
    if g 2 > 11
    then g 3
    else g 4
end
```

3.
```
let fun f 1 = [1]
      | f n = n::(if n mod 2 = 0
                  then f (n div 2)
                  else f (3*n + 1))
in
    f 10
end
```

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

1.
```
fun f  x y z = case x z of
                 0 => ""
               | n => y n
```

2.
```
fun f g (x,[]) = []
    | f g (x,a::l) = case x a of
                       LESS => a::f g(x,l)
                     | EQUAL => f g(x,l)
                     | GREATER => (g a)::f g(x,l)
```

   (Reminder, the type of LESS/EQUAL/GREATER is order)

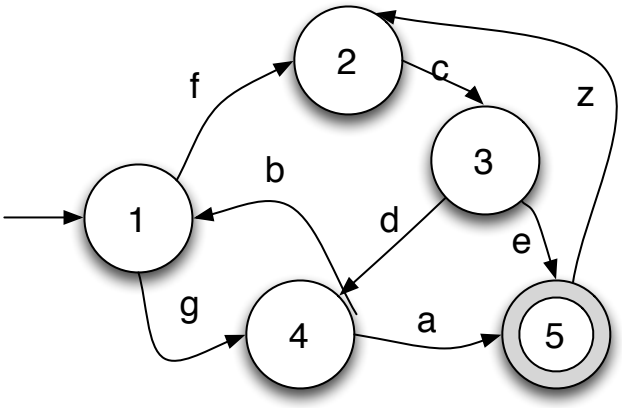
Question 2 Regular Languages [5 pts]

1. Write a regular expression for all strings of `a`s and `b`s in which the substring `aba` appears before the first occurrence of the substring `bb` (both must appear at least once).

2. Write a regular expression for all strings of `0`s and `1`s which does not contain an odd number of consecutive `0`s.

Question 3 NFA to Regexp [10 pts]

Convert the following NFA to a regular expression :

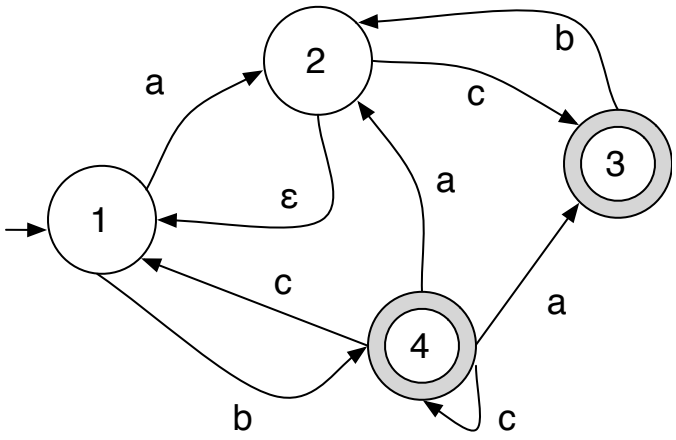


Workspace for question 3

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. (((ab)|(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 (((ab)|(c*d))e*)f

2. The language of sequences of `(`s, `{`, `}`, and `)`s that are properly matched, where `()`s may not appear inside of curly braces (so, `({}({}){})` is in the language, but, `({({})})` is not in the language, even though it is properly matched since the `()` appear inside of `{}`).

Question 7 LL Parsing [10 pts]

Consider the following grammar:

```
0: S -> A $
1: A -> B x C y
2:    | D z B x
3: B -> x B z
4:    | y
5: C -> b C
6:    |
7: D -> C D
8:    | C
```

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

- What are the FIRST and FOLLOW sets of A, B, C and D?

	First	Follow
A		
B		
C		
D		

- Suppose you were writing an LL(1) parser (*i.e.* based on the "predict" sets we discussed in class). Fill in the table below to indicate which rule the parser should use for a given lookahead token (column) when trying to parse a particular non-terminal (row). If multiple rules are possible, write both (indicating a problem with trying to construct this parser). If the input token indicates an error, write E. The row for S is done for you:

	b	x	y	z	$
S	0	0	0	0	E
A					
B					
C					
D					

Question 8 LR Parsing [10 pts]

Consider the following grammar:

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

Below is the state table (`.grm.desc` file) from `ml-yacc`, with the non-error actions replaced by blanks (error actions are removed for space). 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). Two entries (c and S) in State 0 and and all of State 7 are done for you:

```
state 0:
        S : . X

        a       ____________________________
        c       ___shift 2_________________

        X       ____________________________
        S       ___goto 7__________________
state 1:
        S : X .

        .       ____________________________

state 2:
        X : c .

        .       ____________________________
state 3:
        X : a . X X b

        a       ____________________________

        c       ____________________________

        X       ____________________________
state 4:
        X : a X . X b

        a       ____________________________

        c       ____________________________

        X       ____________________________
```

```
state 5:
    X : a X X . b

    b    ____________________________

state 6:
    X : a X X b .

    .   ____________________________

state 7:
   EOF   accept
```

Question 9 Types [10 pts]

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

2. Suppose I have a type system with subtyping, and a top type ($\top$, such that $\forall T.T \sqsubseteq \top$) and a bottom type ($\bot$, such that $\forall T.\bot \sqsubseteq T$). I have two functions, `f:` $\top \to \bot$ and `g:` $\bot \to \top$.

   (a) Describe the set of types that can be passed as the argument to `f`.

   (b) Describe the set of types that can be passed as the argument to `g`.

   (c) Suppose I have another function, `h` and the call `h f` is legal. What can you say about the set of types which can be passed to `h`?

   (d) Suppose I have yet another function, `m` and the call `m g` is legal. What can you say about the set of types which can be passed to `g`?

   (e) (EXTRA CREDIT, probably hard). Suppose I have the following SML declarations:

```
datatype 'a r = MU of 'a r -> 'a

fun y f =
    let fun g (MU x) =  f (fn v => (x  (MU x) v))
    in
        g (MU g)
    end
```

   What is the type of `y`?