Compilers Spring 2014
Midterm Exam

Name: ______________________________

There are 8 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. You may use 1 sheet of notes you created, but no other external resources.

#	**Question**	**Points Earned**	**Points Possible**
1	SML		5
2	Regular Langauges		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 = 10
    val b = 22
in
  a + (if (b < 7) then 99 else 19)
end
```

2.
```
let val a = 7
    val b = let val a = 1
                val q = 2
            in
              [a,q]
            end
    val c = case b of
            [] => (1,0)
          | x::y => (x, 27)
in
   c + a
end
```

3.
```
let fun f (lst, 0) = lst
      | f (lst, b) = f ((b*2+1)::lst, b-1)
in
    f ([143],5)
end
```

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

1.

```
fun f x y = if x then 7 else y -9
```

2.

```
fun f (a::l1,b::l2) = (a,b)::f(l1,l2)
  | f ([],[])    = []
```

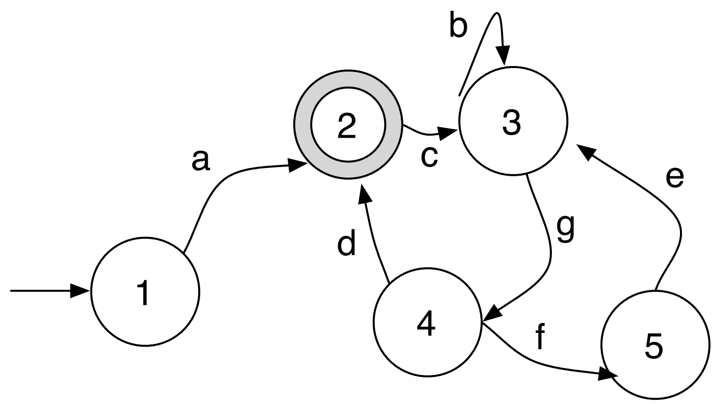
Question 2: Regular Languages [5 pts]

1. Write a regular expression for all strings of `a`s and `b`s in which every `a` must be followed (immediately) by an odd number of `b`s (2 point).

2. Write a regular expression for all strings of `x`s and `y`s where `y`s never occur in pairs (there must be at least one `x` in between). (3 points).

Question 3: NFA to Regexp [10 pts]

Convert the following NFA to a regular expression :

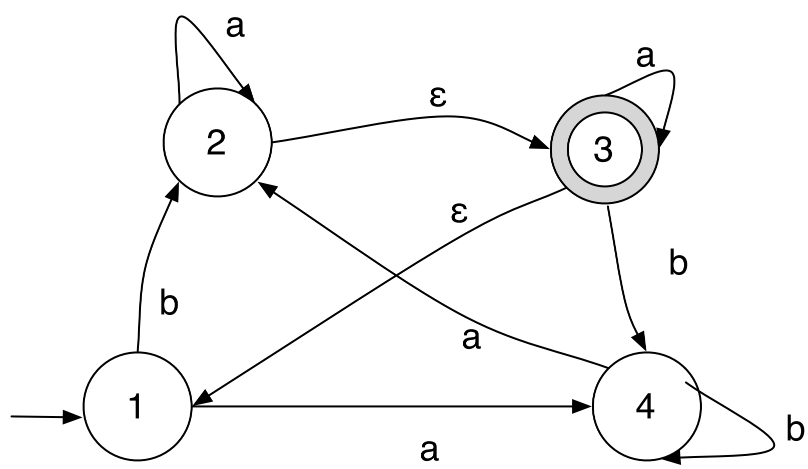


Workspace for question 3

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 (b|c) d? (2 points)

2. a(b|c)* (2 points)

3. ((((ab)|c)*d)—(ef))*g (6 points)

Question 6: Context Free Languages [5 pts]

Write a Context Free Grammar for each of the following:

1. The language described by the regular expression $((ab)^*(c|d))^*$ (2 points).

2. Scheme S-expressions, which are either an identifier, or an open parentheis, followed by 1 or more S-expressions, followed by a closed parenthesis (for example, `((a b) c (d e f))` or `a` or `(c)` (3 points).

Question 7: LL Parsing [10 pts]

Consider the following grammar:

```
S -> a X b
   | b Z a
X -> X c
   | b
Y -> Y X d
   | Z
Z -> c
   |
```

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

- What are the FIRST sets of S, X, Y, and Z? (1 point)

.

- What are the FOLLOW sets of S, X, Y, and Z? (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 language, 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 -> a X c
2: X -> X X
3: X -> b
```

1. What is the start state (set of items) for the state LR parsing state machine for this grammar? (2 points)

2. What is Goto({X $\rightarrow$ .X X }, X)? (2 points)

3. Show the execution of the parser on the string `a b c a b c`. The state machine for the parser is provided along with a table for you to fill in on the next page (6 points).

Using the grammar from the previous page:

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

And the state machine for that grammar:

State	a	b	c	$	S	X
1	s2	s3			g5	g4
2	s2	s3				g6
3	r3	r3	r3	r3		
4	s2	s3		r0		g7
5				Acpt		
6		s3	s8			g7
7	r2	r2	r2	r2		
8	r1	r1	r1	r1		

Fill in the table to the right. In one line, show the current status of the parser—the position in the input, the state the parser is in, and the contents of the stack. In the next line, show the action that the parser takes. Then show the new status in the following line. Repeat this process until the parser accepts the input. The first one is done for you.

Input	**State**	**Stack**
.a b c a b c	1	
Shift to state 2		
a . b c a b c	2	a_1
a b c a b c		
a b c a b c		
a b c a b c		
a b c a b c		
a b c a b c		
a b c a b c		
a b c a b c		
a b c a b c		
a b c a b c		
a b c a b c		
a b c a b c		
a b c a b c		
a b c a b c		
a b c a b c		
a b c a b c		
a b c a b c		

Question 9: Types [10 pts]

1. Show the typing derivation for the Tiger statement `x := f(r.a) + 3`. You may assume that your initial environment (Γ_0) has the following mappings (in addition to the base Tiger environment):
   $\Gamma_0(\text{x}) = \text{int}$
   $\Gamma_0(\text{a}) = \text{int}$
   $\Gamma_0(\text{r}) = \text{Record(a:string, b:int)}$
   $\Gamma_0(\text{f}) = \text{string} \rightarrow \text{int}$

2. Suppose that SML had sub-typing (ignore the fact that it would introduce significant complexities with type inference). What would the sub-typing rule for a `ref` be? That is, under what conditions is a `T ref` a subtype of a `S ref`?

   Hint: think about the two things you can do to a `ref`.