DEPT of ECE, NSU, CSE/CSC 326
SPRING Semester 2015
Midterm Exam

ID **TIME : 1 hour 15 minutes**

NAME **Total Points : 30**

Answer the following questions.

1.[8 points] Write a grammar that generates C like variable declaration statements. Assume variables can have data types int, float, char, or double. Also no initialization of variables are allowed during declaration, i.e., `int a=0;` is not a valid statement. Following is valid variable declaration:

```
int a;
float d,f;
```

Give a parse tree for the above string using your grammar.

We can have a list of declaration statements and each statement is followed by a **;**

So we add productions

Decl_list -> Decl_stmt ; | Decl_list Dec_stmt;

Now Decl_stmt s should start with a type and then followed by a list of IDs.
So, we add productions

Id_list -> ID | Id_list, ID
Type -> int | float | char | double

Now, instead of the parse tree I'm giving the rightmost derivation

Dec_list => Dec_list Dec_stmt ; => Dec_list Type ID_list; => Dec_list Type ID_list, ID;
=> Dec_list Type ID, ID; => Dec_list float ID,ID; => Dec_stmt; float ID, ID;
=> Type ID_list; float ID, ID; => Type ID; float ID,ID; => int ID; float ID,ID;

2. [6 points] Find the First and Follow of the non-terminal S, A, B, C, and D for the following grammar

S → **0**S | A**1**, A → BCD | ∈ , B → **2**S | **1**S | ∈ , C → **3**S | ∈ , D → **4**S

First(S) = { 0, 1, 2, 3, 4} , First(A) = { 1, 2, 3, 4, ε}, First(B) = { 1, 2, ε}, First(C) = { 3, ε}, First(D) = {4}

Follow(S) <- $
Follow(S) <- Follow(B), Follow(C), Follow(D)
Follow(A) <- First(1)
Follow(B) <- First(CD)- ε
Follow(B) <- Follow(A)
Follow(C) <- First(D)
Follow(D) <- Follow(A)

So, Follow(S) = {$, 1, 3, 4}, Follow(A) = { 1}, Follow(B) = { 3, 4, 1}, Follow(C) = { 4}, Follow(D) = {1}

3. [4 points] Show that the following grammar is ambiguous

A → BC | B, B → aB | b, C→ a| ∈

A=> BC => bC => b and we also have A=> B => b. Therefore, the given grammar is ambiguous.

4. [6 points] Using subset construction algorithm convert the following NFA to an equivalent DFA. The NFA states are {1,2,3,4}, start state = 1, final states = {2,4}

States/input	a	b	∈
1	1,2	1	
2	3	3	
3	4	4	4
4			

Start state of the DFA, A = ε-closure({1}) = {1}
Now, move(A, a) = ε-closure($move_{NFA}$({1},a})) = {1,2} = B
move(A, b) = ε-closure($move_{NFA}$({1},b})) = {1} = A
move(B, a) = ε-closure($move_{NFA}$({1,2},a})) = {1,2,3,4} = C
move(B, a) = ε-closure($move_{NFA}$({1,2},b})) = {1,3,4} = D
move(C, a) = ε-closure($move_{NFA}$({1,2,3,4},a})) = {1,2,3,4} = C
move(C, b) = ε-closure($move_{NFA}$({1,2,3,4},b})) = {1,3,4} = D
move(D, a) = ε-closure($move_{NFA}$({1,3,4},a})) = {1,2,4} = E
move(D, b) = ε-closure($move_{NFA}$({1,3,4},b})) = {1,4} = F
move(E, a) = ε-closure($move_{NFA}$({1,2,4},a})) = {1,2,3,4} = C
move(E, b) = ε-closure($move_{NFA}$({1,2,4},b})) = {1,3,4} = D
move(F, a) = ε-closure($move_{NFA}$({1,4},a})) = {1,2} = B
move(F, b) = ε-closure($move_{NFA}$({1,4},b})) = A

States B, C, D, E, and F are final states of the DFA.

5.[6 points] Given the following LL(1) parsing table show how parsing of the string `int*int` would look like. Here, E is the start symbol of the grammar.

	int	*	+	(	)	$
E	E → TX			E → TX		
X			X → +E		X→∈	X→∈
T	T → int Y			T→ (E)		
Y		Y→ *T	Y→∈		Y→∈	Y→∈

stack	input	output
E$	(int*int)$	
↑ stack_top		
E$	(int * int)$	
TX$	(int*int)$	E-> TX
(E)X$	(int*int)$	T -> (E)
E)X$	int * int)$	
TX)X$	int * int)$	E -> TX
int YX)X$	int * int)$	T -> int Y
YX)X$	* int)$	
*TX)X$	* int)$	Y -> *T
TX)X$	int)$	
int YX)X$	int)$	T -> int Y
YX)X$	)$	
X)X$	)$	Y -> ε
)X$	)$	X -> ε
X$	$	
$	$	X -> ε