COSC 542/445 Test #2 WINTER 2016 Name:_____

Open book, notes, computer, Internet.
Do not contact another human for any reason: excepting the professor.
Return your answers on hard copy or digital copy (via email or upload to web) before 1:30pm

For each problem, I have underlined the specific question you are to answer.

1. Execute a shift reduce parser.

Grammar productions:
0) S → R $
1) R → R y
2) R → x

Action and GoTo tables

State	Action			GoTo	
	x	**y**	**$**	**S**	**R**
0	s3				1
1		s4	acc		
2					
3	r2	r2	r2		
4	r1	r1	r1		

The input string is: x y y $

The first three steps of the shift reduce parser are given. Complete the trace of the parser.

STACK	input	action	production
0	x y y $	s3	
0 x 3	y y $	r2	R -> x
0 R 1	y y $	s4	
0 R 1 y 4	y $		

2. Consider the grammar productions (also given in #1):

```
0)   S   →   R $
1)   R   →   R y
2)   R   →   x
```

The initial item in the canonical collection that is used to make the Action and GoTo tables is:
`S → * R $`

2. a. Give the complete item set (just a single set of items) that is obtained by performing closure on

`S → * R $`

2. b. Give the complete item set (just a single set of items) that is obtained from applying `R` to the item that is your answer to 2.a. Include closure on your answer.

2.c. Give the complete canonical set (all item sets). Your answers to 2a and 2b should be a part of this answer.

You can submit your answer as a drawing, or as a text listing of each item by filling in the following table.

Note, the correct answer may require more or fewer rows of this table.

Note also, this question does not require you to give the labeled transitions between item sets.

Item set #	**items**
CC0	`S` → `* R $` …

3. Consider this attribute grammar for evaluating base-10 numbers.

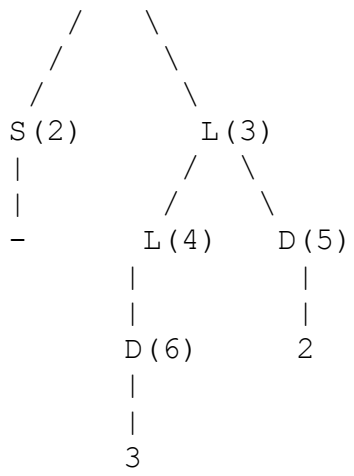
	Production	**Attribution Rules**
1	N → S L	L.position = 0 if S.negative then N.value = - L.value else N.value = + L.value
2	S → +	S.negative = false
3	S → -	S.negative = true
4	L -> D	D.position = L.position L.value = D.value
5	L_0 → L_1 D	L_1.position = L_0.position + 1 D.position = L_0.position L_0.value = L_1.value + D.value
6	D → 0 D → 1 D → 2 D → 3 D → 4 D → 5 D → 6 D → 7 D → 8 D → 9	D.value = 0 * $10^{D.position}$ D.value = 1 * $10^{D.position}$ D.value = 2 * $10^{D.position}$ D.value = 3 * $10^{D.position}$ D.value = 4 * $10^{D.position}$ D.value = 5 * $10^{D.position}$ D.value = 6 * $10^{D.position}$ D.value = 7 * $10^{D.position}$ D.value = 8 * $10^{D.position}$ D.value = 9 * $10^{D.position}$

For each node of the following parse tree (next page), give the final value of the attributes:

L.position, L.value
S.negative
D.position, D.value

Each node is labeled with a number, e.g. (3). You can use the node label to give a textual answer in list form.

N (1)


S(2)
L(3)
L(4)
D(5)
D(6)
2
3

4. Consider this code fragment

```
(1)  a[] = {5, 4, 3, 0};
(2)  for (i = 0; i < 3; i++) {
(3)     if ( i%2 == 0 )
(4)        a[i] = a[i] * 2;
(5)     else
(6)        a[i] = a[i] * 3;
(7)     a[3] = a[3] + 1;
(8)  }
```

4.a. Give the data dependence graph for variable `i`.
You can use the adjacency matrix representation to show the graph, or you can draw a picture.

4.b. Give the control flow graph for the code fragment.
You can use the adjacency matrix representation to show the graph, or you can draw a picture.

5. Consider this code fragment in a C-like language – blocks can be nested.

```
int a, b;

void main() {
   int a;

   {                                        // start block A
      int b, c;

      {                                     // block B
         int c, d;

      }                                     // end B

      /*********************/

      {                                     // start block C
         int a, d;

          {                                 // start block D
             float a, c;

          }                                 // end D


      }                                     // end C

      {                                     // start block E
         int x, y;

      }                                     // end E

   }                                        // end A

}                                           // end main
```

Fill in the information that is recorded in the symbol table giving variable names and scope level. There may be more or fewer rows required than are given here.
Scope levels should be assigned in the manner used in the textbook (i.e., nested scope increases by one).

Name	**Scope level**
a	0
b	0