National University of Computer and Emerging Sciences, Lahore Campus

	Course Name:	Compiler Construction	Course Code:	CS-402
	Program:	BS (CS)	Semester:	Fall 2018
	Duration:	150 Minutes	Total Marks:	50
	Paper Date:	1-Jan-2019	Weight	
	Section:		Page(s):	2
	Exam Type:	Final		

Student : Name:_____ Roll No._____
Section:_____

Instruction/Notes: Solve Q1 on page 1 and 2, Q2 on page 3, Q3 on page 4, and Q4 on page 5. Only the first five pages will be marked!

Question 1 (20 marks)

Consider the following HTML unordered list:

```
<ul>
      <li>Coffee</li>
      <li>Tea</li>
      <li>Milk</li>
</ul>
```

a. Give all the token-lexeme pairs for the above example.
b. Give regular definition for all the tokens.
c. Give transition diagrams (DFA's) for all the tokens.
d. Give a generic CFG for such lists. Do not hardcode to the above example.

Question 2 (10 marks)

Consider a virtual machine that executes three-address code. All variables are global and are stored in a data section. The only data type available is Integer. Following is its code skeleton:

```
int *ds = new int[..]; // data section
int quad[..][4]; // three-address code stored in quadruple
int pc = 0; // program counter
...
for (int pc = 0; quad[pc][0] != HALT; ++pc) {
      switch (quad[pc][0]) {
            case '+': ...
            case '-: ...
            case SWAP: // Add code here!
            case GOTO: // Add code here!
```

```
            ...
        }
    }
```

The machine supports several instructions. Give C/C++ code for the following two instructions:

```
SWAP X, Y
GOTO I
```

The first instruction interchanges the contents of X and Y. While the second instruction jumps to address I.

Question 3 (10 marks)

Consider the following translation scheme that computes the value of a boolean expression:

```
BE -> BE1 or BT  {BE.v = BE1.v || BT.v}
BE -> BT         {BE.v = BT.v}

BT -> BT1 and BF {BT.v = BT1.v && BF.v}
BT -> BF         {BT.v = BF.v}

BF -> true       {BF.v = 1}
BF -> false      {BF.v = 0}
BF -> ( BE )     {BF.v = BE.v}
```

Remove left recursion from the above translation scheme. Do not disturb the precedence and associativity!

Question 4 (10 marks)

A translator converts a C-structure declaration into an equivalent SQL CREATE statement. Consider the following structure for example:

```
typedef struct {
     int id;
     char name[25];
     char address[256];
} Student ;
```

For this declaration, the translator generates the following output:

```
CREATE TABLE Student (
     id int,
     name varchar(25),
     address varchar(256)
);
```

Following is a grammar for such a translator:

```
S -> typedef struct { L } id ;
```

```
L -> L D ; | D ;
D -> int id | char id [ num ]
```

Add required semantic actions into the above CFG for the translation.