National University of Computer and Emerging Sciences, Lahore Campus

Course:	Compiler Construction	Course Code:	CS-402
Program:	BS (CS)	Semester:	Fall 2017
Duration:	180 Minutes (3 Hours)	Total Marks:	60
Paper Date:	22-Dec-17	Weight	
Section:	CS	Page(s):	2
Exam:	Final		

Instruction/Notes: 1) Solve question 1 on page 1-2, question 2 on page 3-4, and so on. Only the first eight pages will be marked!
2) Your work should be neat, clean and easy-to-understand!

Q1 (5+5+5)

Suppose we need to translate personal information of some persons from English to Urdu. Following are some example inputs:

`Name: Ahsan Raza ;`	`Name: Sadia Asif ;`
`Gender: Male ;`	`Gender: Female ;`
`Date of birth: 29-Feb-1985 ;`	`Date of birth: 10-Nov-1990 ;`
`Email: ahsan_raza@gmail.com ;`	`Email: sadia.asif.90@mu.edu.ck ;`

a) Give all the token-lexeme pairs for the first example input (in the order).
b) Give regular definitions for all those tokens which can have more than a single lexeme.
c) Give a CFG for the afore-mentioned translator.

Q 2 (5+10)
a) Give three-address code for the following C++ code:

```
int n, sum, i;
cin >> n;
sum = 0;
i = 1;
for (i = 1; i <= n; ++i)
      sum = sum + i;
cout << sum;
```

b) Consider the following CFG for the C++ "for" loop:

```
S -> for ( S ; BE ; INC ) S
S -> id = E
S -> { L }
L -> L ; S
L -> ^
BE -> id ro id
INC -> ++ id
```

Now add semantic actions into the above CFG, to generate three-address code. You are also required to give actions for the assignment statement, the increment, and the boolean expression. However you need not to provide actions for the arithmetic expression. For simplicity, assume the increment can be of one type only.

Q3 (5+5+10)

Consider the following translation scheme:

```
G -> <graph> L </graph>     {G.t = L.t}
L -> $L_1$ E                   {L.t = $L_1$.t + "," + E.t}
L -> $ E                    {L.t = "#" + E.t}
E -> <edge> N $N_1$ </edge>    {E.t = "(" + N.t + "," + $N_1$.t + ")"}
N -> <node> num </node>     {N.t = num.lex}
```

a) Now give parse tree (without semantic actions) for the following graph:

```
<graph>
     <edge>
          <node>1</node> <node>2</node>
     </edge>
     <edge>
          <node>1</node> <node>3</node>
     </edge>
</graph>
```

b) Write output of the above translation scheme for the given graph.

c) Remove left recursion from the above translation scheme.

Q4 (5+5)

a) Consider the following Lex code:

```
%{
#include <stdio.h>
#include <stdlib.h>
int sum = 0;
%}

str (letter|digit)*
letter [A-Z|a-z]
digit [0-9]

num digit*

%%
```

```
{str} { printf("%s \t", yytext); }

{num} {sum = sum + atoi(yytext);}

{"\n"} { printf("%d \n", sum);
     sum = 0; }
%%

int main() {
     yylex();
     return 0;
}
```

What will be the output of the generated translator for the following input:

Yasir	50	60	70
Asad	60	70	80
Zain	70	80	90

b) Consider the following grammar:

```
S -> 0 S 1
S -> ^
```

Now show working of a bottom-up parser for the following string: `0011`