Ministry of Higher Education			Acad. Year	:	2020 / 2021
El-Shorouk Academy			Term	:	Second
Higher Institute for Computer & Information Technology			Year	:	4th
			No. Questions	:	4
Department :	Computer Science		Date	:	24/06/2021
Instructor :	Dr. Ahmed elabbassy		Time Allowed	:	2 Hours
Reviewer1 :	Dr. Abd-Alatif Hussin		Max. Marks	:	80 Marks
Reviewer2 :	Dr. Salah Eliwa		No. of pages	:	2 pages

4203 Compiler Theory (Final Exam)

Answer all of the following questions

Question 1 (20 points)

1.1 Write regular expressions for the following, or give reasons why no regular expressions can be written:

a) All strings of a's and b's which contains the substring: abba

(a|b)* abba (a|b)*

b) Integers are sequences of digits, possibly preceded by the minus sign (-).

(-)?digit+

c) All strings of digits that represent odd decimal numbers.

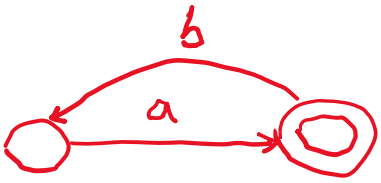
digit*(1|3|5|7|9)

d) Strings over the set of alphanumeric characters where there are exactly as many letters as digits.

There is no regular expression

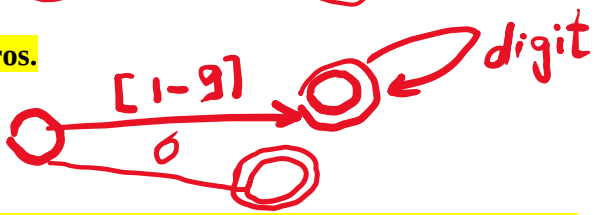
1.2 Draw a DFA that accepts the following:

a) The regular expression: a(ba)*

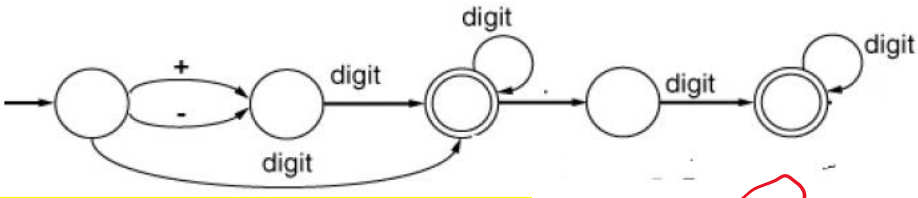


b) All strings of digits without leading zeros.

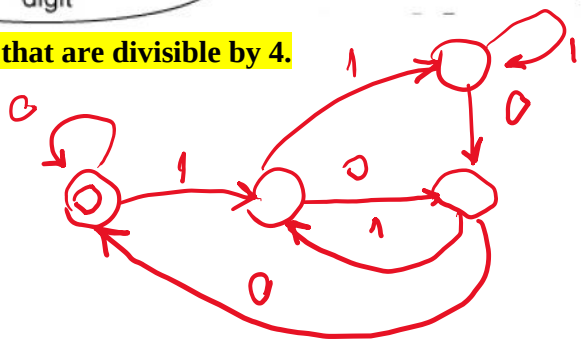
[1-9]digit* | 0



c) Floating-point numbers are sequences of digits containing a decimal point, possibly preceded by the minus sign.



d) Binary numbers that are divisible by 4.

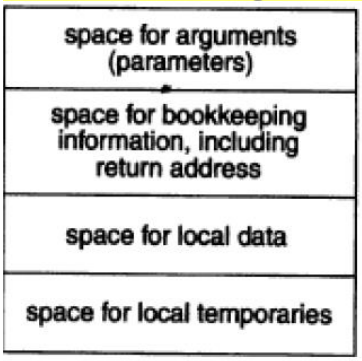


empty string is considered as 0

1.3 Describe what must be done on a return from a function call.

- Sequence of operations performed when return from procedure calls
  - Find the arguments and pass them back to the caller.
  - Free the callee environment.
  - Restore the caller environment, including PC.

1.4 Describe the organization of activation record for a function.



Question 2 (20 points)

2.1 At what phases are regular expressions used in compilers?

The scanner (Lexical Analysis)

2.2 When is a context free grammar unambiguous?

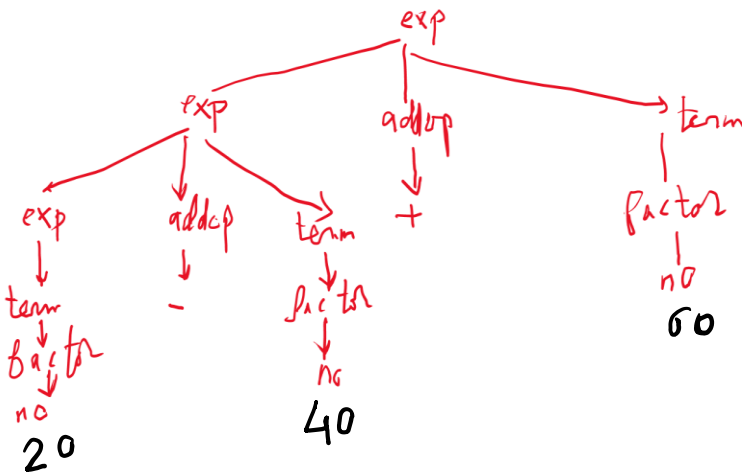
The grammar to permit a string to have only one parse tree

2.3 Given the following BNF grammar:

exp → *exp addOp term* | *term*
addOp → + | -
term → *term multOp factor* | *factor*
multOp → *
factor → *(exp)* | number

Write a leftmost derivation for the expression 20 - 40 + 60 and draw its parse tree.

(1) *exp* => *exp addOp term*
(2) => *exp addOp term addOp term*
(3) => *term addOp term addOp term*
(4) => *factor addOp term addOp term*
(5) => number *addOp term addOp term*
(6) => number - *term addOp term*
(7) => number - *factor addOp term*
(8) => number - number *addOp term*
(9) => number - number + *term*
(10) => number - number + *factor*
(11) => number - number + number

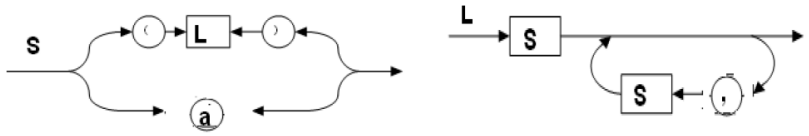


2.4 Consider the following grammar:

$$S \rightarrow (L) \mid a$$
$$L \rightarrow L , S \mid S$$

a) Translate the grammar into EBNF and draw its syntax diagrams.

$$S \rightarrow (L) \mid a$$
$$L \rightarrow S \{, S\}$$



b) Write pseudo code to parse the above EBNF grammar using recursive-decent.

```
procedure S
begin
  case token of
  ( : match( ( );
    L;
    match( ));
  a:
    match (a);
  else error;
  end case;
end S
```

```
procedure L;
    begin
        S;
        while token = , do
            match(token);
            S
            end while;
    end L;
```

```
procedure match( expectedToken);
begin
 if token = expectedToken then
   getToken;
 else
   error;
 end if;
end match
```

Question 3 (20 points)

3.1 Give two examples of syntax errors and two examples of semantic errors.

Syntax errors

Using "=" when "==" is needed.

```
if (number=200)
```

Missing semicolon

```
int a = 5          // semicolon is missing
```

Errors in expressions:

```
x = (3 + 5;  // missing closing parenthesis ')'
```

Semantic Errors cLanguage

```
extern int foo(int a,int b,int c,int d);
int fee()
{
int f[3],g[1],h,i,j,k;
char *p;
foo(h,i,"ab",j,k);
k = f*i+j;
h = g[17];
printf("%s,%s\n",p,q);
p = &(i+j);
k = 3.14159;
p = p*2;
}
```

3.2 What is the difference between assembler and compiler?

Assembler is A translator for the assembly language of a particular computer which is a symbolic form of one machine language (Low level)

A compiler is a translator for a high level language like c, c++

3.3 Given the following BNF grammar:

$$declaration \rightarrow type\ var\text{-}list;$$
$$type \rightarrow \text{int} \mid \text{float} \mid \text{bool}$$
$$var\text{-}list \rightarrow \text{identifier} , var\text{-}list \mid \text{identifier}$$

a) Left factor this grammar.

$$declaration \rightarrow type\ var\text{-}list$$
$$type \rightarrow \text{int} \mid \text{float} \mid \text{bool}$$
$$var\text{-}list \rightarrow \text{identifier}\ \ var\text{-}list$$
$$var\text{-}list' \rightarrow , var\text{-}list \mid \varepsilon$$

b) Construct First and Follow sets for the non-terminals of the resulting grammar.
You are asked to show the computation process.

First set

Grammar Rule	Pass 1	Pass2
declaration → type var-list ;		First (declaration)= {int, float}
type→ int	First(type)= {int}	
type→ float	First(type)={int, float}	
var-list→ identifier *var-list'*	First (var-list)= {identifier}	
var-list'→ , var-list	First (var-list')={ ,}	
var-list'→ ε	First (var-list')={ , ε }	

Follow set

Grammar Rule	Pass 1
declaration → type var-list ;	Follow (declaration)= {$} Follow (type)= {identifier} Follow (var-list)= {$} ∪ {;}
var-list→ identifier *var-list'*	Follow(var-list') = {$} ∪ {;}
var-list'→ , var-list	

c) Construct the LL(1) parsing table for the resulting grammar.

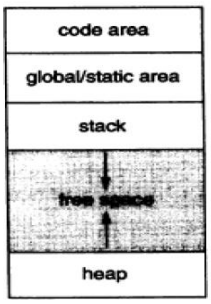
Parsing table

M[N,T]	identifier	,	int	float	$
declaration			*declaration → type var-list ;*	*declaration → type var-list ;*	
type			*type→* int	*type→* float	
var-list	*var-list→* identifier *var-list'*				
var-list'		*var-list'→ , var-list*			*var-list'→* ε

d) Show the action of corresponding LL(1) parser given the input string: bool x , y;

Step	Parsing Stack	Input	Action
1	decl$	bool x,y; $	decl → type varlist;
2	type varlist;$	bool x,y; $	type → bool
3	bool varlist;$	bool x,y; $	Match
4	varlist;$	x,y; $	varlist →id varlist'
5	id varlist';$	x,y; $	Match
6	varlist';$	,y; $	varlist' →, varlist
7	,varlist;$	,y; $	Match
8	varlist;$	y; $	varlist →id varlist'
9	id varlist';$	y; $	Match
10	varlist';$	; $	varlist' → ε
11	;$	; $	Match
12	$	$	Accept

3.4 Draw a diagram illustrating the Memory Organization during program execution.



Question 4 (20 points)

4.1 Briefly describe the purpose of the syntax and semantic analysis phases in a compiler.
compute additional information needed for compilation that is beyond the capabilities of Context-Free Grammars and Standard Parsing Algorithms

Syntax Analysis
Takes the sequence of tokens produced by the scanner as its input and produces the syntax tree as its output. It Determines the syntax, or structure, of a program using the grammar rules of a

4.2 Define annotated parse tree.

A parse tree augmented with the attribute values at each node is called an annotated parse tree: The syntax tree annotated with attributes

4.3 Explain how to describe semantic using syntax directed semantic definitions.

Syntax Directed Definitions

- Use the syntactic representation of language to drive semantic analysis.
- A syntax directed definition is an extension of a context free grammar in which Each grammar symbol is assigned certain attributes
- Each production is augmented with semantic rules which are used to define the values of attributes
- The process of computing the attributes of each node is called *annotating* the parse tree, which is then called an *annotated* or *attributed* parse tree (or syntax tree).

41

4.4 Briefly describe the purpose of dependency graphs.

- **The attribute equations indicate the order constraints on the computation of the attributes.**
  - **Attribute at the right-hand side must be computed before that at the left-hand side.**
  - **The constraints are represented by directed graphs — dependency graphs.**

- **"Dependency graphs"** are a useful tool for determining an evaluation order for the attribute instances in a given parse tree.

- While an annotated parse tree shows the values of attributes
  - A dependency graph helps us determine how those values can be computed.

4.5 Given the following arithmetic expression:

if (d < 5) then x = 3 else x = 5;

a) Write down the corresponding three-address code.

```
t1= d<5
if_f t1 goto L1
x=3
goto L2
LABEL L1
X=5
LABEL L2
……..
```

b) What is the quadruple representation for this code?

```
(lt, d, 5, t1)
(if_f, t1,L1, _)
(asn,3,x,_)
(jmp,L2,_,_)
(LABEL,L1,_,_)
(asn,5,x,_)
(LABEL,L2,_,_)
-------
```

<<Good Luck>>