تجميعات اختبارات سابقه جزئية المد الأول

Q1) Put a circle around the symbol of the best answer for each of the following

1. Which of the following features must be available in a programing language to be suitable for writing embedded systems
   a) Provide constructs for low-level control over hardware.
   b) Use floating point representation.
   c) Have features to analyze data.
   d) All of the above.

2. An orthogonal programing language
   a) Has a relatively small set of primitive constructs that can be combined in a relatively small number of ways to get the desired result.
   b) Has a fewer number of exceptions because every possible combination is legal.
   c) Is good with respect to reliability.
   d) All of the above.

3. Which of the following features a programing language need to have in order to be good with respect to reliability
   a) Data types.
   b) Support for abstraction.
   c) Exception handling.
   d) All of the above.

4. Having a small number of manageable features and constructs, makes a programming language good with respect to
   a) Readability.
   b) Writability.
   c) Reliability.
   d) All of the above.

5. A language with too many operators and special symbols is
   a) Good with respect to writability
   b) Bad with respect to reliability
   c) Bad with respect to readability
   d) All of the above.

6. Java is partly compiled and partly interpreted (hybrid implementation) makes it
   a) Good for writing portable programs.
   b) Good for writing efficient programs.
   c) Good for reducing the cost of training.
   d) All of the above.

7. Classes, inheritance and polymorphism designed in programing languages as a
   a) The Von-Neumann architecture.
   b) Programing methodologies.
   c) People preferences.
   d) None of the above.

Q2)

A) Which programming language is better with respect to reliability Java or C++? Explain why?

List features or constructs makes it so (Explain why)

Different question same concept and answer (Compare between Java and C++ with respect to reliability).

B) Which programming language is better with respect to the cost of execution Java or C++? Give at least 2 reasons.
List features or constructs makes it so (Give at least 2 reasons)?

C) Which programming language is better with respect to portability Java or C++? Explain why?
List features or constructs makes it so (Explain why)?

D) Discuss two advantages of C++ compared to Java.

E) Discuss how pointers in C++ affects

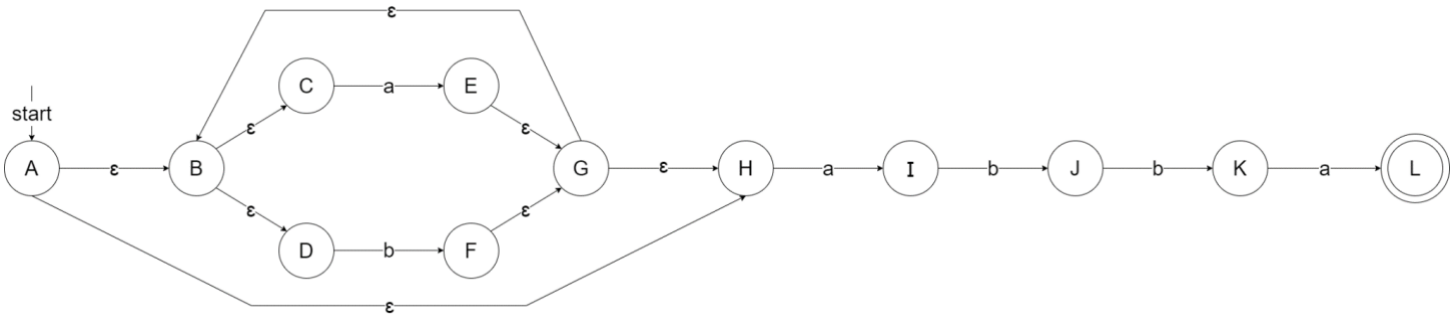
1)writability

2)reliability

F) Why were modular languages developed?

G) Discuss how Von Neumann architecture influenced the design of modern programming languages.

Q3) Consider the following NFA.



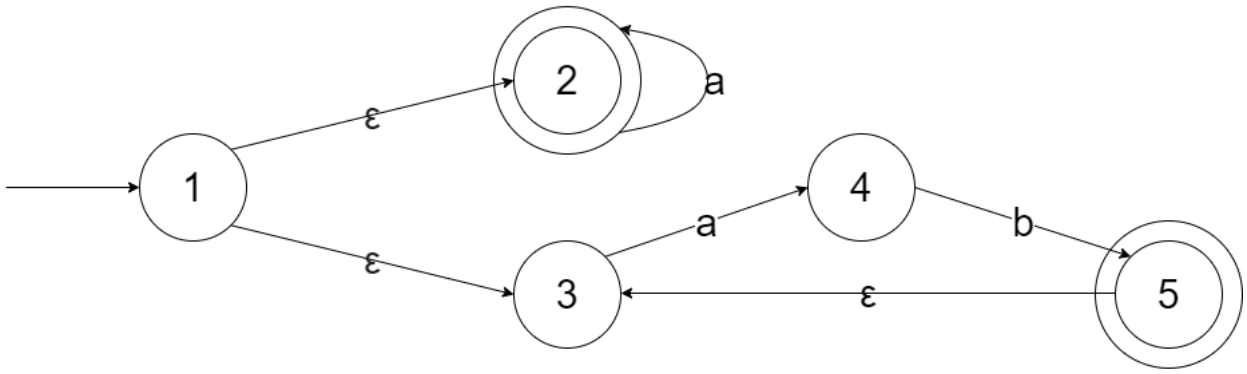
A) Describe the language accepted by this NFA using a regular expression.

B) Convert the above NFA into a DFA.

C) Represent the DFA using a table.

D) Write an algorithm that uses the above table to decide if a string is acceptable by the DFA or not.

Q4) Consider the following NFA



A) Describe the language recognized by the NFA using regular expression.

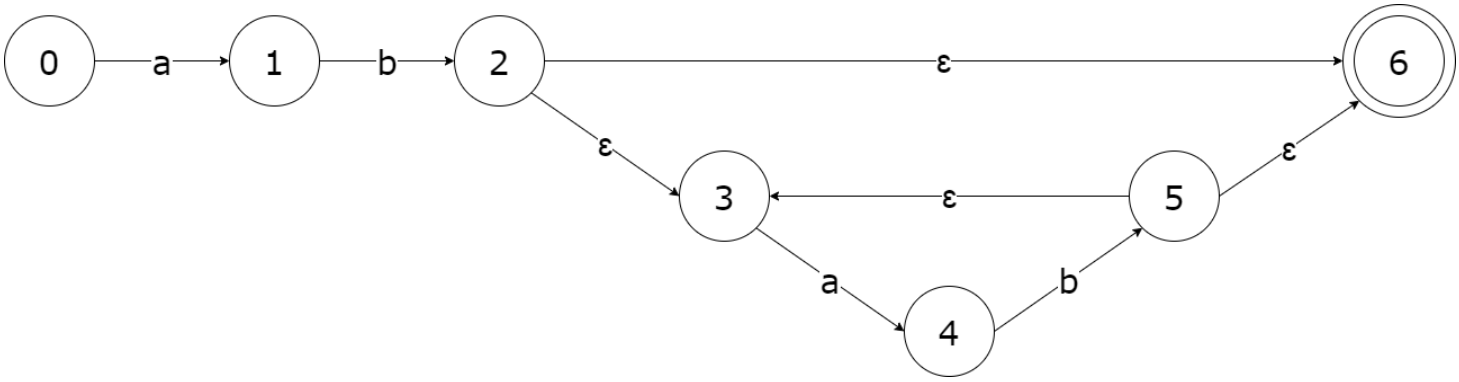
B) Convert the above NFA into a DFA.

C) Represent the DFA using a table.

D) Write an algorithm that takes a string over the alphabet and display “accept” if the string is acceptable by the DFA, or “reject” if it is not.

Convert the regular expression $(0+1)^+$ over the alphabet {0,1} into an NFA.

Q5) Consider the following NFA.



A) Describe the language accepted by the above NFA using regular expression.

B) Convert the above NFA into a DFA.

Q6) Consider the following DFA over the alphabet ∑ = {a,b}

A) In your own words describe the language accepted by the above DFA.

B) Write a regular expression that describes the language accepted by the above DFA.

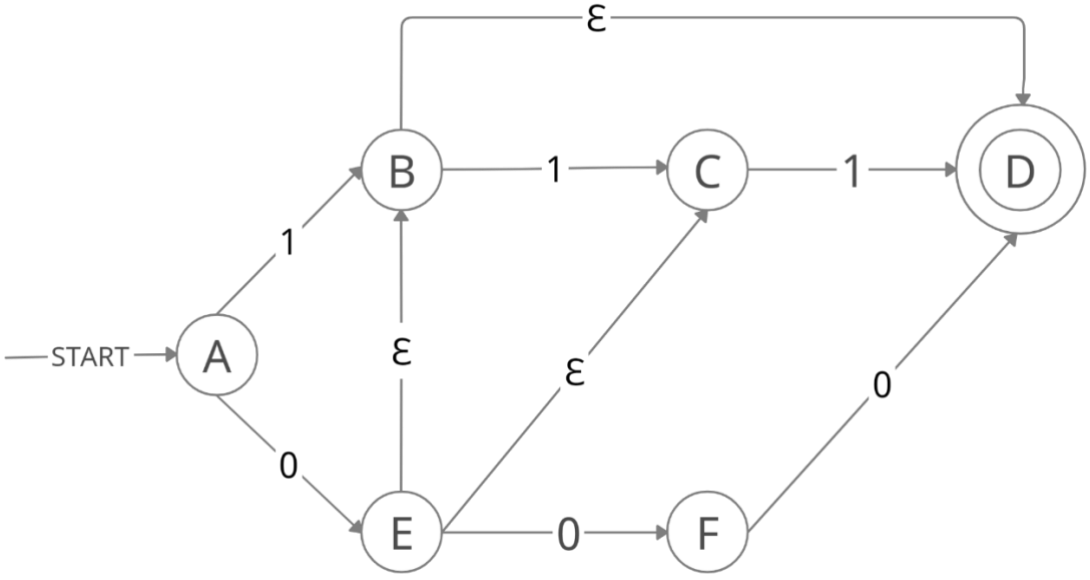
C) Represent the DFA using a table.

D) Write an algorithm that uses the above table to decide if a string is acceptable by the DFA.

Q7)

A) Draw an NFA that recognizes the language denoted by the regular expression 1(0+1*).

B) Write a regular expression that describes the language recognized by the NFA.



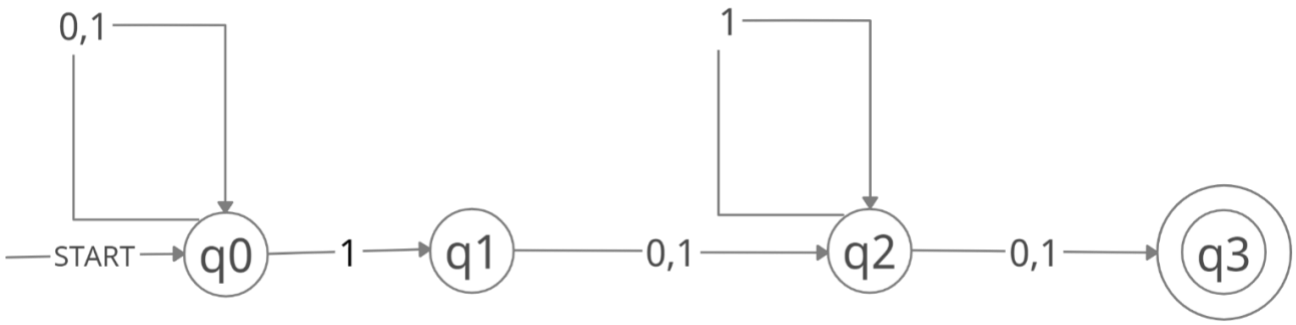
C) Represent the above NFA using a table.

D) Write an algorithm that uses the above table to determine if a given string is legal.

Q8)

A) Design a DFA that accepts a string if it belongs to the language $\{a^p b^q c^r \mid p,q,r \geq 0\}$

B) Consider the finite automation in the following figure.



1. What is the set of reachable states for the input string 0011?

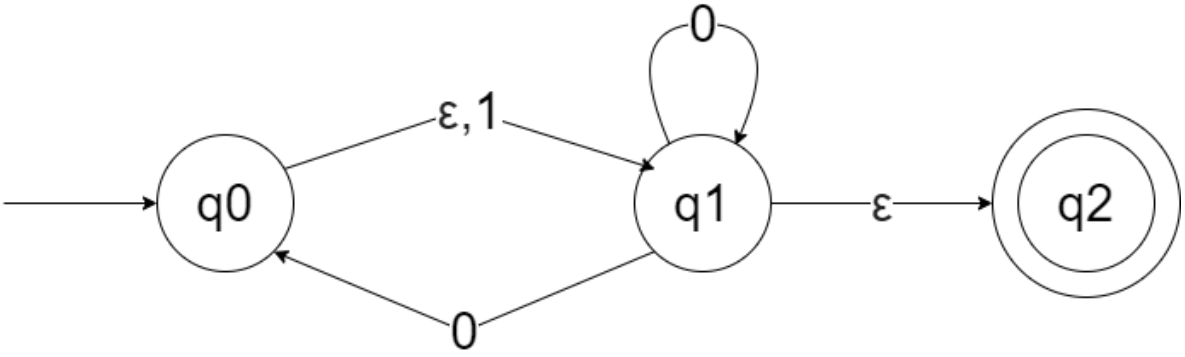
2. Is the string 0011 acceptable? Why?

C) Represent the above finite automaton using a table.

Q9)

A) Convert the regular expression below into an NFA.
$(a+b)^*aab^*$

B) Consider the following NFA



1. Represent the NFA using a table.

2. What is the shortest string that would be rejected by the above NFA.

3. Write an algorithm that accepts a string if it is accepted by the above NFA.

4. Convert the above NFA into a DFA.

Q10) Consider the alphabet S={a,b,c}.

A) Design a DFA that accepts a string in which every ‘a’ is followed by a ‘b’ but no ‘b’ is followed by ‘c’. for example, the DFA should accept “cabbab” but not “abc”.

B) Write a regular expression that describes the language accepted by the above DFA.

Q11)

A) Design a DFA that accepts a string containing letter and digits where the string consists of at most two characters the first character must be a letter (A-Z), while the second may be a letter or digit (0-9).

B) Write a regular expression that defines the token class identifier where an identifier can consist of at most two characters the first character must be a letter (A-Z), while the second may be a letter or digit (0-9).

Q12) consider the following CFG.

E→ E – int | E / int | int

A) Which operation has the highest precedence?

B) What is the association of the "−" operation?

C) What is the association of the "/" operation?

D) Show that the string int-int/int belongs to the language of the above CFG (is valid)?

E) Write a CFG that describes all mathematical expression that can be formed using only * and + . Make sure that * has a higher precedence over + and the associativity of both operation is left to right.

Q13) Write a CFG describing a language over the alphabet ∑={a,b}, containing all string with

A) Equal number of a's and b's.

B) The number of a's is double the number of b's.

Q14)

A) Assuming that a float number consists of an optional integer part and an optional fraction part, write a regular expression that describes the language of all float numbers, note that the string "14.15", ".15" and "15." Are all legal but the string "." Is not.

B) Write a CFG that describes the language of float numbers as described above.

C) Consider a language defined over S=[0,1] where a string belongs to the language if the number of 1's is twice the number of 0. What would you use to describe this language regular expression of context free grammar? Justify your answer.

تجميعات اختبارات سابقه جزئية المد الثاني

Q1)

A) consider the following CFG

E→ E+E | int

1) Show that the grammar is ambiguous.

2) Rewrite the grammar to resolve the ambiguity in such away that + has the left associativity.

B) Rewrite the following CFG so that we can write recursive descent parser for E→ E+E | E*E | id.

C) Rewrite the following CFG so that we can write an LL(1) for E→ id+E | id*E | (E) | id.

D) Write a recursive descent parser in pseudo code (or c or java) for the grammar

E→ id+E | id*E | (E) | id

Q2)

A) Consider the following CFG

E→ E+T | T

T→ T*F | F

F→ (E) | id

1) Find the first and follow set of

Symbol	First set	Follow set
E		
F		
)		

B) Assume that a grammar contains the production E→ T+F | TF.
Assume also that first(T)={id,ε}, first(F)={*, ε}, follow(E)={id,$} and follow(T)={+,$}

1) Place the production E→ T+F and E→ TF in the right entries of the LL(1) table.

	Id	*	+	$
E				

Is the table LL(1)? Why? Or why not?

Q3) Consider the following grammar

E→SX

X→aE | ε

S→(E) | cY | ε

Y→bS | ε

Find the first and follow set for each symbol in the table below

Symbol	First set	Follow set
E		
X		
S		
Y		
(		

Q4) Consider the following CFG (please notice that it is slightly different than the previous question)

S→ S*E | E+S

E→ int | ε

A) Find the first and follow set for each terminal and non-terminal symbol.

Symbol	First set	Follow set
S		
E		
Int		
*		
+		

B) Draw the LL(1) parsing table

	Int	*	+	$
S				
E				

C) Is the grammar LL(1)? Why or why not?

D) If your answer to the above question was no, then rewrite the grammar so that the grammar is LL(1).

Q5) Consider the following CFG grammar

E→int*E | int | ε

A) Draw the LL(1) parsing table for the above grammar.

	Int	*	$
E			

B) Is the above grammar an LL(1) grammar.

C) If your answer above was “yes” then explain why, and if it was “no” then explain why and rewrite the grammar to make it an LL(1) grammar.

Q6) consider the following CFG

E→ int*E | int+E

E→ int

A) Write two left most derivation for the string int*int+int*int
First Derivation:

Second Derivation:

B) Since the string int*int+int *int has more than one leftmost derivations, can we conclude that the grammar is ambiguous.

C) Which operation has the highest precedence? Why?

D) Is the associativity of the operation * right or left associativity?

E) Rewrite the above grammar so that you reverse the * operation has the higher precedence than the + operation and both operation have right associativity.

Q7) Assuming that there is a state in the DFA for recognizing the viable prefixes of a grammar contains the items.

E→ T.+F

T→F.

1) Under what condition would the SLR algorithm reduce?

2) Under what condition would the SLR algorithm shift?

3) Assuming the follow(T)=follow(F)={+,*,$}, is the grammar SLR? Why or why not?

Q8) Consider the following CFG

S→ S*E | E+S

E→ int | ε

A) Design an NFA that recognizes the viable prefixes for the above grammar.

B) Use the following LR(0) DFA to parse the string (id, id+id)$ using the SLR parsing algorithm.
Please use a suitable table to illustrate your answer.

C) Is the CFG represented using the above DFA an SLR grammar? Why?

D) Give two examples of the problems that cannot be detected by both the lexical and the parser.

Q9) Consider the following CFG

$S' \rightarrow E$

$E \rightarrow int*E \mid int+E \mid \varepsilon$

A) Design an NFA that recognizes the viable prefixes of the grammar.

B) Convert the above NFA to a DFA.

C) Is the above grammar an SLR grammar?

D) If your answer above was “yes” then explain why, and if it was “no” then explain why
and rewrite the grammar to make it an SLR grammar.

Q10) Draw the symbol table when line 14 in the following code is processed.

```
0: int x = 137;
1: int z = 42;
2: int MyFunction (int x, int y) {
3:      printf("%d,%d,%d,\n", x, y, z);
4:      {
5:          int x, z;
6:          z = y;
7:          x = z;
8:          {
9:              int y = x;
10:             {
11:                 printf("%d,%d,%d,\n", x, y, z);
12:             }
13:         printf("%d,%d,%d,\n", x, y, z);
14:         }
15:     printf("%d,%d,%d,\n", x, y, z);
16:     }
17:}
```

Q11)

A) List three examples of errors that cannot be detected by lexical analyzer nor by the parser.

B) Fill in the rectangles below to make the following inference rules useful for a type checking system.

1)

f is and identifier.

f is non-member function in scope S.

f has type $(T_1 , \ldots . , T_n) \rightarrow U$

$$\frac{\square}{S \vdash f(e_1 , \ldots . , e_n) : U}$$

2)

$$\frac{S \vdash e_1 : T[] \qquad S \vdash e_2 : \text{int}}{S \vdash e_1[e_2] : \square}$$

تجميعات اختبارات سابقه جزئية الفاينل

Q1) Put a circle around the symbol of the best answer for each of the following

1. The main advantage of interpreters over compilers is
    a) efficiency
    b) portability
    c) reliability
    d) a+b

2. Java programs are partly compiled and partly interpreted to achieve
    a) portability
    b) efficiency
    c) portability and efficiency
    d) reliability

3. Which of the following programming language features affects the reliability of the programming language
    a) Data types
    b) Exception handling
    c) Orthogonality
    d) All of the above

4. Which of the following programming language features affects the writability of the programming language
    a) Data types
    b) Exception handling
    c) Orthogonality
    d) All of the above

5. Which of the following regular expressions describe the language over the alphabet {a,b} that consists of all strings that contain at least one 'a'.
    a) $b^*aa^*b^*$
    b) $b^*a(a^*b^*)^*$
    c) $b^*ab^*a^*b^*$
    d) none of the above

6. Which of the following regular expressions describe the language over the alphabet {a,b} that consists of all strings that begin and end with the same letter.
    a) ab*a+ba*b
    b) a(b*a)*+b(a*b)*
    c) ab*a*b*a+ba*b*a*b
    d) none of the above

7. Converting an NFA that consists of n states to an equivalent DFA would result in a DFA that contains
    a) exactly 2^n states
    b) at least 2^n states
    c) at most 2^n states
    d) n! states

8. A table that represents an NFA of 3 states and deals with 2 terminal symbols consists of
    a) 3 rows and 2 columns
    b) 3 rows and 3 columns
    c) 2 rows and 3 columns
    d) 5 rows and 2 columns

9. Which of the following methods require eliminating left recursion
    a) Recursive descent parsing
    b) LL(1) parsing
    c) a+b
    d) Bottom up parsing

10. Which of the following methods may require left factoring a grammar
    a) Recursive descent parsing
    b) Bottom up parsing
    c) LL(1) parsing
    d) a+b

11. Which of the following productions assigns left associativity for +
    a) E→ int+E
    b) E→E+E|int
    c) E→E+int
    d) None of the above

12. Which of the following grammars is not ambiguous
    a) E→ id+E | id*E | (E) | id
    b) E→E+E | E * E | (E) | id
    c) E→E+T|T
       T→ T*F | F
       F→ (E) | id
    d) a+c

Question 13-16 are related to the following grammar

E→E+T|T
T→ T*F | F
F→ (E) | id

13. What is the follow set of E?
    a) { + }
    b) { $, +,) }
    c) { +,) }
    d) { $, + ,), ε }

14. What is the follow set of F?
    a) { + }
    b) { $, +,),*}
    c) { +,) }
    d) { $, + ,), ε }

15. What is first set of E?
    a) { (}
    b) { (, id }
    c) {id}
    d) { (, id, ε }

16. In the LL(1) table for the above grammar, the entry that contains E+T is
    a) row E column (
    b) row E column id
    c) row E column $
    d) a+b

The following assumptions are related to questions 17-19

Assume that a grammar contains the production E→T+F|TF.

Assume also that first(T)={id, ε}, first(F)= { * , ε }, follow(E)={id,$} and follow(T)={+, $}

17. In the corresponding LL(1) table, which entry contains TF
    a) row E, column id
    b) row E, column +
    c) row E, column *
    d) a and b

18. which entry in the LL(1) table must contain ε
    a) row E, column id
    b) row E, column +
    c) a and b
    d) none of the above

19. The grammar is not LL(1) grammar because the entry at
    a) row E column $ is multiply defined
    b) row E column id is multiply defined
    c) row E column + is multiply defined
    d) the grammar is LL(1) because no entry is multiply defined

20. Bottom-up parsers are
    a) More efficient than top-down parser
    b) More general than top-down parser
    c) Less general than top-down parser
    d) a and b

The following assumptions are related to questions 21-25

Assuming that the DFA for recognizing the viable prefixes of a grammar contains the items
E→T.+F
T→F.

21. Which types of conflicts does the above grammar contain
    a) Reduce-reduce conflict
    b) No conflicts
    c) Shift reduce conflict
    d) A+b

22. The grammar is not SLR(1) grammar if
    a) follow(T)={+}
    b) follow(E)={+,$}
    c) first(F)={+}
    d) none of the above

23. If the top of the stack contains + and the next input token is +, then the SLR parsing algorithm
    a) pops + from the stack
    b) reports and error
    c) moves to the next input token
    d) a+c

24. If the top of the stack contains + and the next input token is *, then the SLR parsing algorithm
    a) pops + from the stack
    b) moves to the next input token
    c) reports and error
    d) a+b

25. Which of the following errors are not detected by a parser
    a) Undeclared identifier
    b) Unexpected data type
    c) Identifier declared more than once
    d) All of the above

Consider the following rule to answer questions 26- 28

$$\frac{\begin{array}{c} f \text{ is an identifier.} \\ f \text{ is a non-member function in scope S.} \\ f \text{ has type } (T_1, \ldots, T_n) \rightarrow U \\ S \vdash e_i : R_i \text{ for } 1 \leq i \leq n \\ R_i \leq ?? \text{ for } 1 \leq i \leq n \end{array}}{S \vdash f(e_1, \ldots, e_n) : ?}$$

26. What should be written in place of the double question marks (??) in the rule to make correct?
    a) U
    b) T_i
    c) f
    d) None of the above

27. What should be written in place of the question mark (?) in the rule to make correct?
    a) f
    b) U
    c) T_i
    d) None of the above

28. Assuming that a primitive type is convertible to itself, then the above inference rule is applicable for
    a) Referenced types
    b) Primitive types
    c) None of the above
    d) a and b

29. In a stack machine, the heap section of the memory is dedicated for
    a) Local variables
    b) Actual arguments
    c) dynamically allocated objects
    d) a and b

30. The memory section allocated to global variables is allocated
    a) Dynamically
    b) Statically
    c) At execution time
    d) None of the above

31. Bottom-up parsers are preferred because they are
    a) more efficient than top-down parsers.
    b) more general than top-down parsers.
    c) simpler than top-down parsers.
    d) All of the above

32. Recursive descent parsers are not used in real compilers because they are
    a) too complicated
    b) require left-factorization
    c) too inefficient
    d) all of the above

33. One of the following operations is not used on symbol tables
    a) push scope
    b) insert symbol
    c) pop symbol
    d) look-up symbol

34. A type error is detected by
    a) The parser
    b) The semantic analyzer
    c) The lexical analyzer
    d) The code generator

35. An SLR parser uses a DFA to determine
    a) If what it is on the stack is a viable prefix of the handle
    b) If what is on the stack is the handle
    c) What to action to execute next shift or reduce
    d) a + c

Q2) Assuming that x is the first argument, write down the code that will be generated (by cgen) for the following expressions (i.e. what would be the output of the compiler?).

a) **3+1**

b) **if x=1 then x else f(1,0)**

Q3)

a) Write the code generator segment that deals with the if statement cgen (if e1=e2 the e3 else e4)

b) What code will be generated for the following procedure? (i.e. what would be produced by the compiler?).

```
Def f(x,y) =

        If x=y then x else 0
```

Q4)

a) Write a regular expression that describes the language of all integers that consist of at most 3 digits with an optional sign (+ or -).

b) Design a DFA that accepts an integers that if it consists of at most 3 digits with an optional sign (+ or -).

c) Give an example of a language that is irregular (not regular).

Q5) Consider the CFG

E→E+E | E*E | int | id

a) Give an example that illustrates that the above grammar is ambiguous.

b) Rewrite the grammar so that * has higher precedence the +.

c) Assuming that ambiguity is fine , rewrite the given grammar so that we can write a recursive descent parser for it.

Q6) consider the following regular expression

(A*+B*)1

a) Convert the regular expression above into an NFA.

b) Convert the NFA above into a DFA

c) Represent the DFA using a table.

d) Write an algorithm (pseudo code) that a string as input and displays “yes” if a string belongs to the language (A*+B*)1: otherwise display “no”.

Q7) Consider the following grammar

E→TX

T→(E) | int Y

X→+E | ε

Y→*T | ε

a) Find the first set and follow set for each terminal and non-terminal symbol used in the grammar.

Symbol	First set	Follow set
T		
E		
X		
Y		
(		
)		
Int		
+		
*		

b) Draw the LL(1) parsing table.

c) Write down the SLR(1) parsing algorithm

d) Assuming dynamic type checking, rewrite the above rules so that they work for the reference types.

$$\frac{S \vdash e_1 : T[] \qquad S \vdash e_2 : \text{int}}{S \vdash e_1 = e_2 : T}$$

Q8)

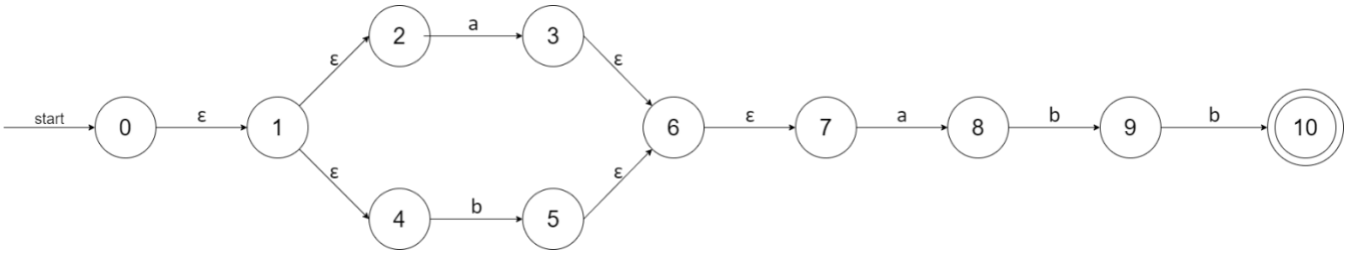
a) Write cgen for the expression e1 + e2

b) Write code that cgen generates for the expression 5+7.

Q9)

a) Why is “null” cases a problem for type systems? And how do they deal with it?

Q10) Consider the following NFA



a) Describe the language of this NFA using a regular expression.

b) Convert the above into a DFA

Q11) Consider the CFG

E→E-E | E+E

E→ int | id

a) Rewrite the CFG in such a way that allows you to write a recursive descent parser for it.

b) Write a recursive descent parser in Java (or as a pseudo code) for the grammar you wrote in above (in a).

c) Rewrite the grammar in such a way that makes an LL(1) parser possible for the grammar.

Q12)

a) What is the main reason that makes parsers unable to detect all errors? Give examples of 4 errors that cannot be detected by a parser.

   Reason:

   Examples:
   1)

   2)

   3)

   4)

b) In your own words, describe what each of the following rules mean.

1)

f is an identifier.
f is a non-member function in scope S.
f has type $(T_1,\ldots\ldots,T_n) \rightarrow U$
$S \vdash e_i : T_i$ for $1 \leq i \leq n$

$S \vdash f(e_1,\ldots\ldots,e_n) : U$

2)

$S \vdash e_1 : T$
$S \vdash e_2 : T$

$S \vdash e_1 = e_2 : T$

c) Assuming dynamic type checking ,rewrite the above rules so that they work for reference types.

1)

2)

Q13) Consider the following DFA that recognizes the viable prefixes for the grammar

E→E’

E→E+T | T

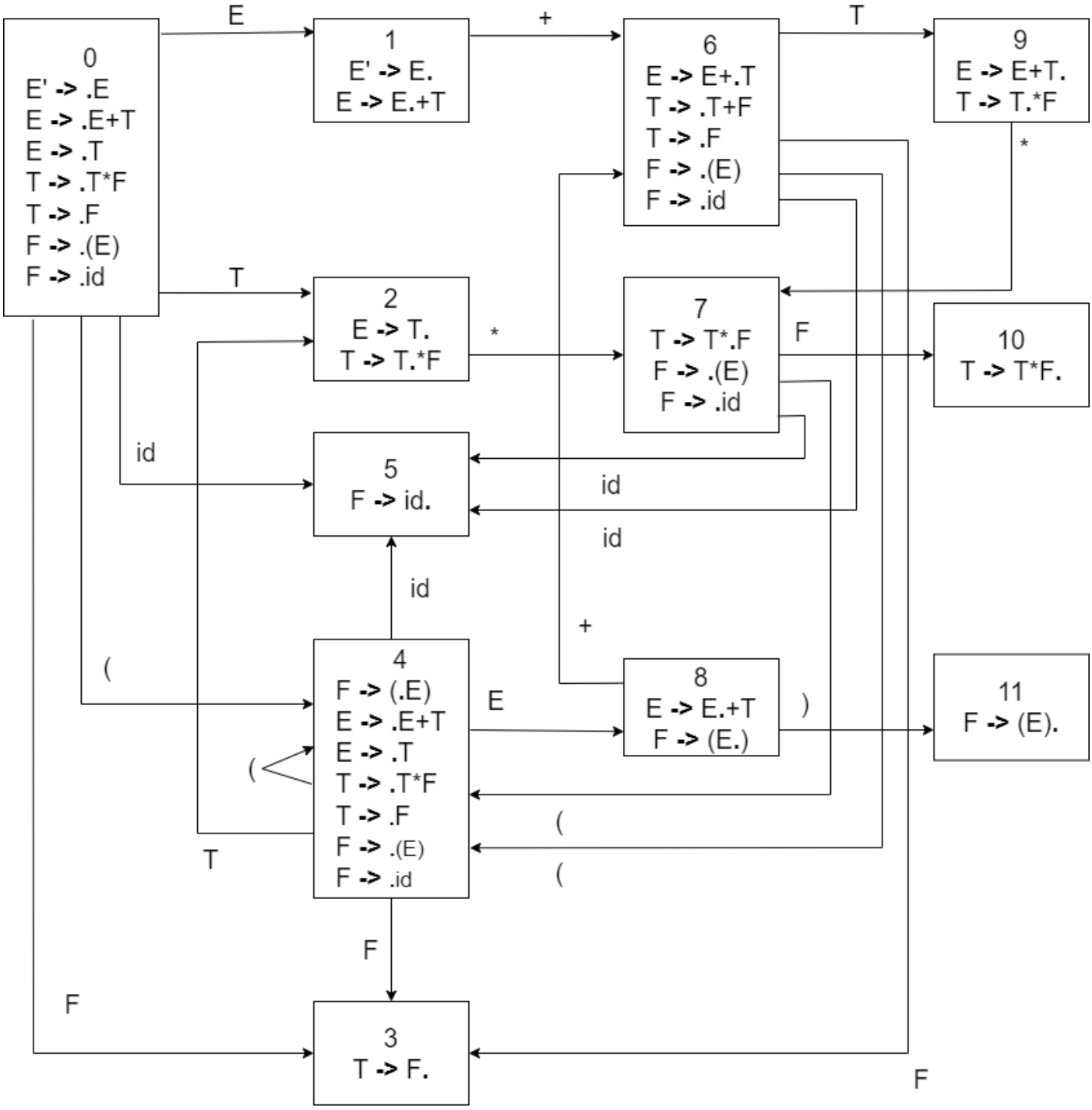
T→T*F | T

F→(E) | id

a) Construct an LL(1) table for the grammar.

b) Is the grammar an LL(1) grammar? Why or why not?

c) The following NFA recognizes the viable prefixes for the above grammar. Is the grammar an SLR(1) grammar? Why or why not?



Q14) Write cgen for each of the following expressions.

1) If e1=e1 then e3 else e4

2) 5

Q15)

a) Write code that cgen generates for the following procedure

b) Write the code that cgen generates for the invocation statement

Q16)

a) Write a regular expression for a language over the alphabet strings that begin and end with a different character.

b) Design a DFA, over the alphabet ∑={a,b} that accepts a string if contains exactly two b's and one a.

c) Convert the regular expression (aa*)+b into an NFA

Q17) Consider the following CFG

X→X+X | X*X | int

a) Show that the grammar is ambiguous.

b) Rewrite the grammar so that + has a left associativity and * has right associativity.

Q18) Consider the following grammar over the alphabet ∑={u,v,w,x,y,z};

S→UVW

U→u | Wv | ε

V→w | xU | ε

W→y | z | ε

a) Complete the missing entries in the following table that gives the first and follow sets for each terminal and non-terminal symbols.

Symbol	First	Follow
S		
U		
V		
W		
u		
v		
w		
x		
y		
z		

b) Draw the corresponding LL(1) table

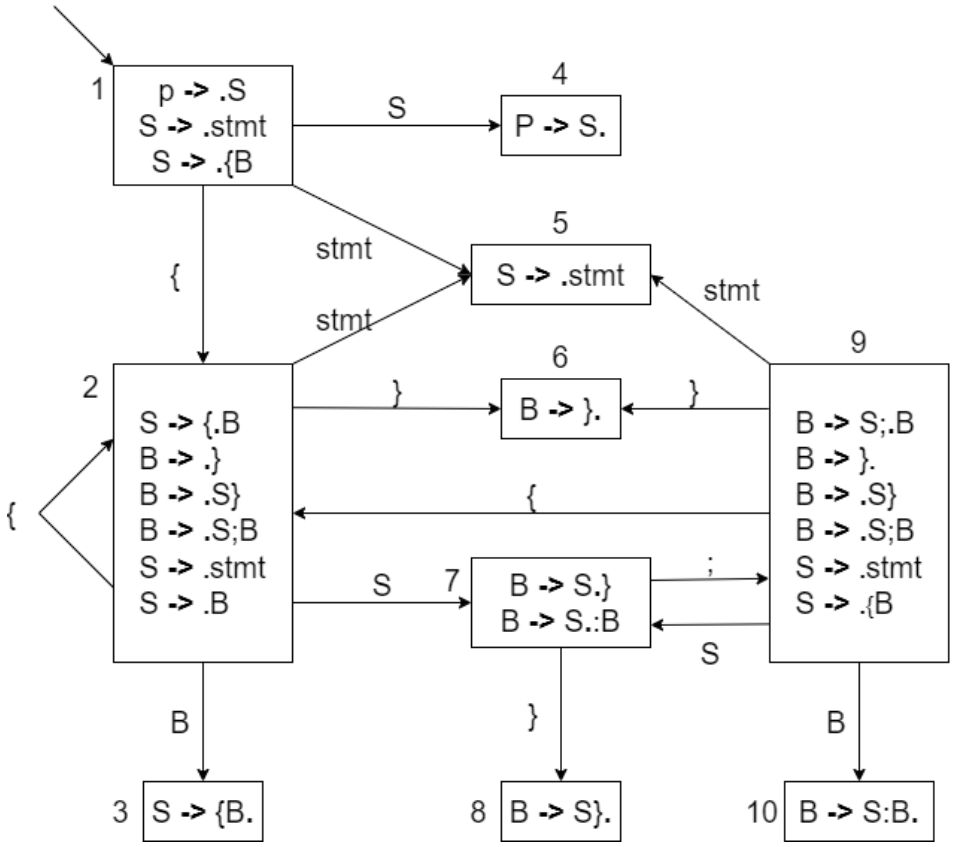
c) Is the grammar an LL(1) grammar? Why or why not?

Q18)

a) Draw an NDF that recognizes all valid items for the grammar
   X→X+X | X*X | int

b) Is the above grammar SLR grammar? Why or why not?

c) Consider the following DFA for a viable prefixes of a grammar



Assuming that an SLR parser resolves shift-reduce conflicts by choosing the shift, show the operation of such a parser on the input string {stmt;;}.
Write your answer as atable of a suitable format and make it clear if the statement is accepted or rejected.

Q19) Consider the following program

```
0: int x = 137;
1: int z = 42;
2: int MyFunction(int x, int y) {
3:      printf("%d, %d, %d\n", x, y, z);
4:      {
5:              int x, z;
6:              z = y;
7:              x = z;
8:              {
9:                      int y = x;
10:                     {
11:                             printf("%d, %d, %d\n", x, y, z);
12:                     }
13:                     printf("%d, %d, %d\n", x, y, z);
14:             }
15:             printf("%d, %d, %d\n", x, y, z);
16:     }
17:}
```

1) Draw the symbol table that you have used when you reached line 13, mark each variable with the line number it was declared in, by writing above it @ followed by the line number.

Q20)

a) Write a code generator for the language constructs
   If e1=e2 then e3 else e4

b) Write the code that would be produced by cgen for the expression x+5. You can assume that x is the only variable in the procedure.