Q1) A) Discuss how does each of the following factor affects the quality of a programming language with respect to readability, writability and reliability. (12 marks)

A. Support for abstraction (3 marks)

1. Readability

it is good with Readability, because it will be easier to read the program without the complex details.

2. Writability

it is good with writability, because you can use functions without knowing the details.

3. Reliability

it is good with Reliability, because if you have a reliable abstraction you will have less errors instead of you (writing which will have more errors.

B. Orthogonality (3 marks)

1. Readability

it is good with Readability, because the fewer the combination the esaier you can read the program.

2. Writability

it is bad with writability, because the fewer the combination the less expressive power you have

3. Reliability

it is good with Reliability, because the fewer the combination the less exceptions happens.

C. Pointers as in C and C++ (3 marks)

1. Readability

Pointers are bad with respect of Readability, because pointer arthmetic makes it harder to read the program and know where the pointer points at.

2. Writability

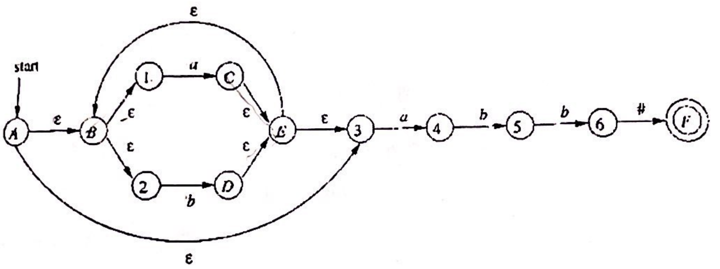
Pointers are good with respect of writabillty, because it provides more expressivenss in writing and it is flexable.

2

3. Reliability

Pointers. are bad with respect of Reliability, because mistakes can happen more easier in pointer arthmetic

Q2 Consider the following NFA (16 marks)



$(a+b)^*abb\#$

1. Describe the language accepted by this NFA using a regular expression. (2 marks)

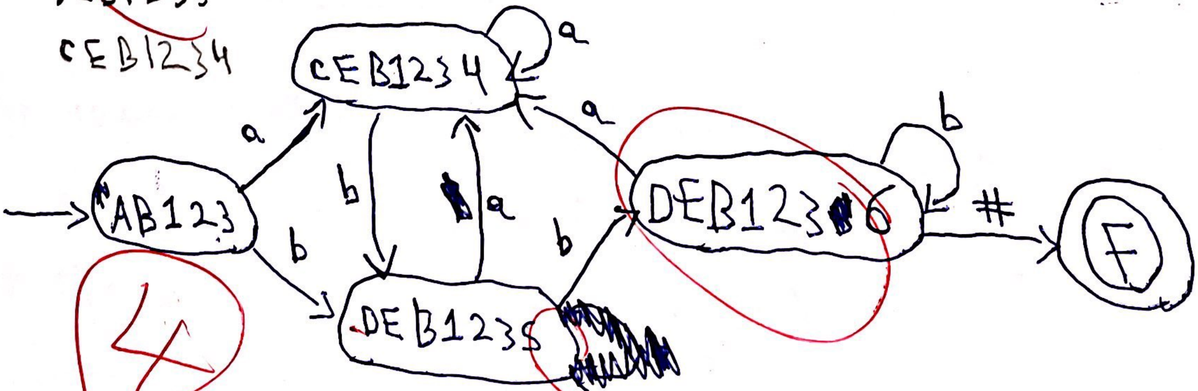
2

$(a+b)^*abb\#$

DEB1235

CEB1234

2. Convert the above NFA into a DFA. (6 marks)



4

bb#
accepted ?

DEB1235 · DEB12356 · DEB1236
CEB1234 · CEB1224 · CEB1234 ·

3. Represent the DFA using a table. (4 marks)

	a	b	#
AB123	CEB1234	DEB1235	—
CEB1234	CEB1234	DEB1235	—
DEB1235	CEB1234	DEB1236	—
DEB1236	CEB1234	DEB1236	F
F	—	—	—

4

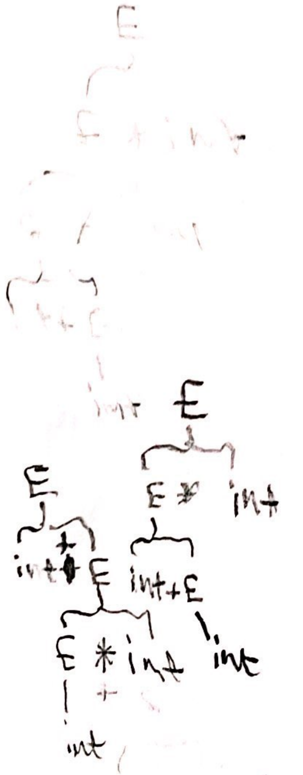
4. Write an algorithm that uses the above table to decide if a string is acceptable by the DFA or not. (4 marks)

```
State = S  \\ S is start state
i = 0
while(input[i]):
    state = Table[state, input[i++]]
end while
if(state == F):        else:
    Accept                 reject
```

4

Q3) Consider the following CFG (12 marks)

E → int | int + E | E * int | (E)



A) Which operator has a higher precedence + or *? (2 mark)

2

it depends on the parse tree because the grammer is ambguis, the leftmost or Right most has higher precedence

B) What is the associativity of the * operation? (2 mark)

~~It depends what the parse tree ends with, if it ends with + operator it is Right to left, if it ends with * operator it is Left to Right~~

the associativty of * operation is Left to Right

2

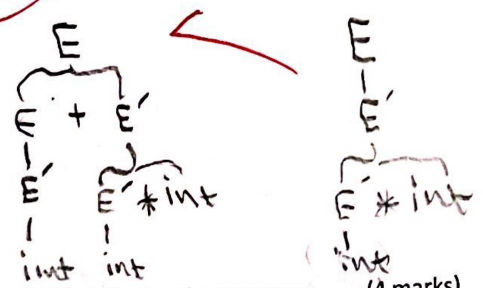
Scanned with CamScanner

C) Rewrite the grammar to ensure that * has higher precedence and all operations have left-to right associativity. (4 marks)

$E \rightarrow E + E' \mid E'$

$E' \rightarrow E' * int \mid (E) \mid int$

4



D) Left factor the grammar. (4 marks)

$E \rightarrow E' + int \mid E' * int \mid (E')$

$E' \rightarrow E \mid int$

0