University of Asia Pacific

Admit Card

Final-Term Examination of Spring, 2021

Financial Clearance	PAID

Registration No : 17201052

Student Name : Jannatun Nayem Jemy

Program : Bachelor of Science in Computer Science and Engineering

SI.NO.	COURSE CODE	COURSE TITLE	CR.HR.	EXAM. SCHEDULE
1	CSE 425	Computer Graphics	3.00	
2	CSE 426	Computer Graphics Lab	1.50	
3	CSE 429	Compiler Design	3.00	
4	CSE 430	Compiler Design Lab	1.50	
5	BUS 401	Business and Entrepreneurship	3.00	
6	BUS 402	Business and Entrepreneurship Lab	0.75	
7	CSE 457	Design and Testing of VLSI	3.00	
8	CSE 458	Design and Testing of VLSI Lab	0.75	
9	CSE 400	Project / Thesis	3.00	

Total Credit: 19.50

1. Examinees are not allowed to enter the examination hall after 30 minutes of commencement of examination for mid semester examinations and 60 minutes for semester final examinations.

2. No examinees shall be allowed to submit their answer scripts before 50% of the allocated time of examination has elapsed.

3. No examinees would be allowed to go to washroom within the first 60 minutes of final examinations.

4. No student will be allowed to carry any books, bags, extra paper or cellular phone or objectionable items/incriminating paper in the examination hall.
Violators will be subjects to disciplinary action.

This is a system generated Admit Card. No signature is required.

Admit Card Generation Time: 15-Nov-2021 11:43 AM

Scanned with CamScanner

University of Asia Pacific
Department of CSE
Final Examination of Spring, 2021

Name: Jannatun Nayem Jemy
Registration No: 17201052
Course Name: Compilet Design
Course Code: CSE 429
Date: 25.11.21

REDMI NOTE 8
Scanned with CamScanner

Ans: to the Q.no: 1 (a)

Three Address code for the above statement:

1. Sum = 0;
2. if (student 1 < student 2) goto 4
3. goto (7)
4. T_1 = number + 1
5. number = T_1
6. goto (9)
7. T_2 = number - 1
8. number = T_2
9. T_3 = sum + 1
10. Sum = T_3
11. if (Sum < 10) go to (2)

REDMI NOTE 8
AI QUAD CAMERA

Scanned with CamScanner

1(b)

Quadraples:

	operation	Arg 1	Arg 2	Result
1	=	0		Sum
4	+	number	1	T_1
5	=	T_1		number
7	−	number	1	T_2
8	=	T_2		number
9	+	Sum	1	T_3
10	=	T_3		Sum

Triple:

	Operation	Arg 1	Arg 2
1	=	0	
4	+	number	1
5	=	4	
7	−	number	1
8	=	7	
9	+	Sum	1
10	=	9	

REDMI NOTE 8

Scanned with CamScanner

1 (c)

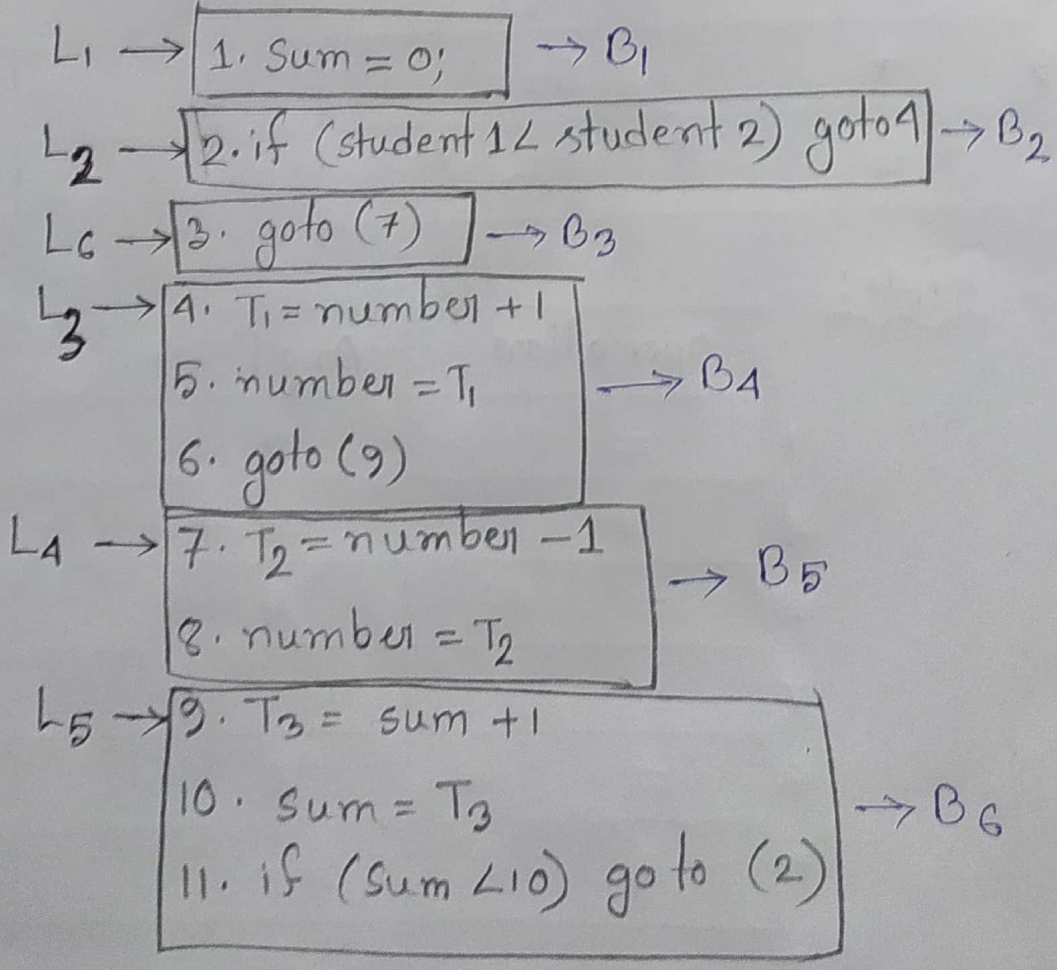
a) Reason for choosing a Leader:

i) First line of code is leader.

ii) Address of conditional and unconditional goto are leader.

iii) Immidiate next line of goto also leader.

Identify leadre of 1(a):



REDMI NOTE 8
AI QUAD CAMERA

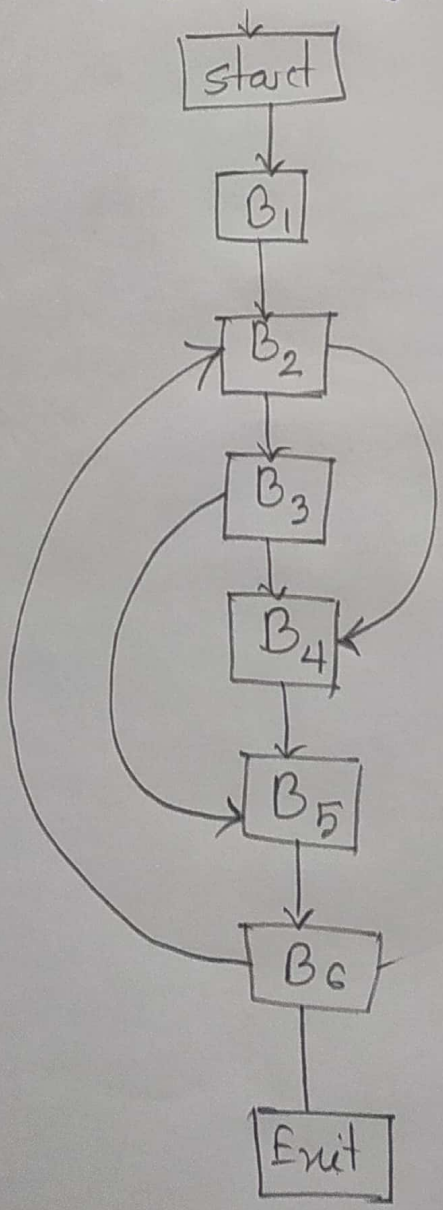
Scanned with CamScanner

1(d)

Basic blocks:

Finding basic blocks in 1(c)

Construct the flow graph:



REDMI NOTE 8
AI QUAD CAMERA

Scanned with CamScanner

2(a)

Given Grammar:

REXPR → REXPR + RTERM | RTERM

RTERM → RTERM RFACTOR a | RTERM RFACTOR b |
RTERM RFACTOR

RFACTOR → RFACTOR * RPRIMARY | RPRIMARY

RPRIMARY → a | b

i) For ~~elim~~ a eliminating:

R TERM → RTERM RFACTOR a

R TERM → RTERM RFACTOR b

~~R TermP~~

R TERM → R-TERM RFACTOR

R TERM → RTERM RFACTOR RTERM

R-TERM' → a | b | ∈

(Ans:)

REDMI NOTE 8
AI QUAD CAMERA

Scanned with CamScanner

ii)

Left factoring eliminate make grammar more suitable for ~~tab~~ top-down parsing because it removes the common left factor that appears in two productions of the same non terminal. It is done to avoid backtracking by the parser. That's why we can say that left factoring eliminating is make suitable for top-down parsing.

iii)

Yes, In the above-stated grammar has left ~~re~~ Recursion.

For eliminating:

① ⇒ REXPR → REXPR + RTERM | RTERM

REXPR → RTERM REXPR'

REXPR' → + RTERM REXPR' | ∈

REDMI NOTE 8
AI QUAD CAMERA

Scanned with CamScanner

(ii) ⇒

RTERM → RTERM RFACTOR a | RTERM RFACTOR b | RTERM RFACTOR

RTERM → RTERM'

RTERM' → RFACTOR a RTERM' | RFACTOR b RTERM' | RFACTOR RTERM' | ∈

(iii) ⇒

RFACTOR → RFACTOR RPRIMARY | RPRIMARY

RFACTOR → RPRIMARY RFACTOR'

R FACTOR' → * RPRIMARY RFACTOR' | ∈

REDMI NOTE 8
AI QUAD CAMERA

Scanned with CamScanner

iv) Eliminating of left Recursion is more suitable for top - down parsing tables, because:

It is a case of when all the left most non terminal in a production of a non-terminal is the non-terminal itself on through some other non-terminal destinations, rewrites to the non-terminal again, It removes looping on top-down parsing.

So, it also helps to make suitable grammar for top- down parsing.

REDMI NOTE 8
AI QUAD CAMERA

Scanned with CamScanner

2(b)

Recursive Decent Parser:

A recursive descent parser is a kind of top-down parser, built from a set of mutually recursive procedures where each such procedure implements one of the nonterminals of the grammer.

Given Grammer:

Exp → Ax

Exp → Bw

A → xx C

A → x x y x

C → y y y

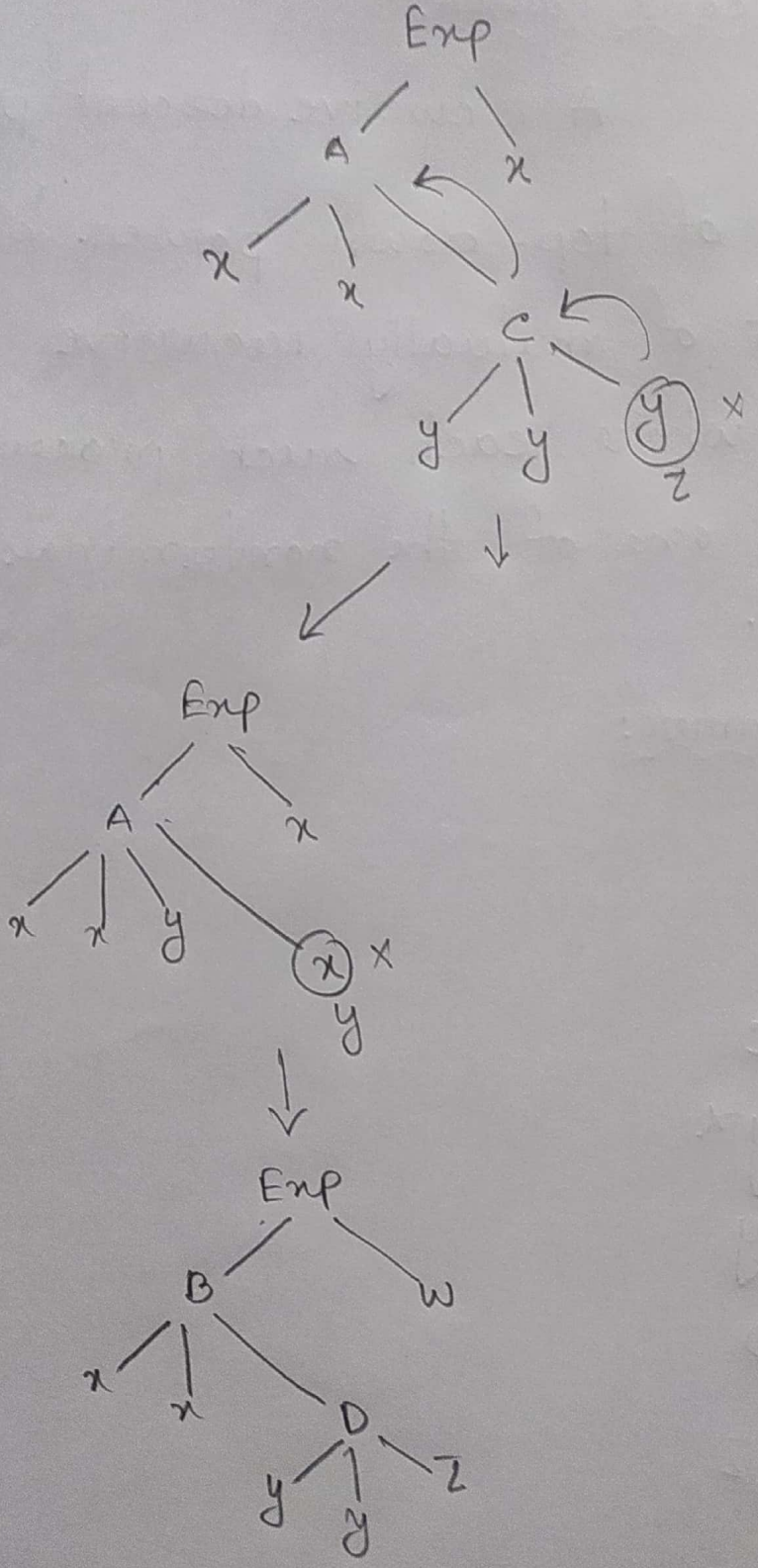
B → x x D

D → y y z

REDMI NOTE 8
AI QUAD CAMERA

Scanned with CamScanner

Input string: xxyyzw



So, this will be for given string.

REDMI NOTE 8
AI QUAD CAMERA

Scanned with CamScanner

ID-17201052

11

3(i)

Calculating First and follow of the give (CFG)

First:

P = { if, while, print, id, int }

P' = { if, while, print, id, int, ϵ }

S = { if, while, print, id, int }

S' = { id, int }

T = { if, while, print, id, int }

T' = { else }

E = { id, int }

Follow:

P = { $ }

P' = { $ }

S = { else, if, while, print, id, int, $ }

S' = { else, if, while, print, id, int, $ }

S'' = { else, if, while, print, id, int, $ }

T = { else, if, while, print, id, int, $ }

T' = { else, if, while, print, id, int $ }

E = { if, while, print, id, int else, $, then }

REDMI NOTE 8

Scanned with CamScanner

3 (ii)

LL(1) parsing table for given grammar:

	if	while	print	then	else	id	int	$
P	P→SP′	P→SP′				P→SP′	P→SP′	
P′	P′→P	P′→P	P′→P			P′→P	P′→P	P′→ϵ
S	S→ifS′	S→while ES	S→print E			S→E	S→E	
S′						S′→ES″	S′→ES″	
S″				S″→then T				
T	T→ST′	T→ST′	T→ST′			T→ST′	T→ST′	
T′					T′→else S			
E						E→id	E→int	

REDMI NOTE 8
AI QUAD CAMERA

Scanned with CamScanner

3(iii)

If we take string:

String: if id then print int
↑ ↑ ↑ ↑ ↑

Method	Stack	Input	Action
	P $	if id then print int$	output → SP'
	SP'$	if id then print int$	output → S → if S'
	if S'P'$	if id then print int$	match if
	S'P'$	id then print int$	output → E S''
	E S''P'$	id then print int$	output S → id
	id S''P'$	id then print int$	output match id
if	S''P'$	then print int$	output S'' → then T P'
if	then TP'$	then print int$	match then
if	TP'$	then print int$	output → S T'
if id	S T' P' $	print int $	output S → Print E
if id	print E T' P' $	print int $	match print
if id then	E T' P' $	print int $	E → int
if id then	int T' P' $	int $	match. int
if id then print	T' P' $	int $	output
if id then print	TP'$	int $	P' → ε
if id then print	P'$	$	
if id then print int	$	$	
if id then print int		$	
if id then print int			

REDMI NOTE 8
Scanned with CamScanner

4(a)

0. $S' \rightarrow S$
1. $S \rightarrow i\ e\ t\ s$
2. $S \rightarrow i\ e\ t\ s\ e\ s$
3. $S \rightarrow w\ e\ do\ s$
4. $S \rightarrow b\ L\ e$
5. $S \rightarrow s$
6. $L \rightarrow L, S$
7. $L \rightarrow S$

Scanned with CamScanner

ID-1720105D2

REDMI NOTE 8
AI QUAD CAMERA

Scanned with CamScanner

ID-1720106Z

REDMI NOTE 8
AI QUAD CAMERA

Scanned with CamScanner

4(ii)

	i	e	t	w	do	b	;	s	$	S	L
I_0	S_e			S_3		S_4		S_5		I_1	
I_1									Acc		
I_2		S_6									
I_3		S_7									
I_4	S_2			S_3		S_4					
I_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5	r_5			
I_6			S_{10}								
I_7					S_{11}						
I_8		S_{12}					S_{13}				
I_9	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7	r_7		
I_{10}	S_e			S_3		S_4		S_5		I_{14}	
I_{11}	S_e			S_3		S_4		S_5		I_{15}	
I_{12}	r_4	r_4	r_4	r_4	r_4	r_4	r_4	r_4	r_4		
I_{13}	S_2			S_3			S_4		S_5	I_{17}	
I_{14}		I_6									
I_{15}	r_3	r_3	r_3	r_3	r_3	r_3	r_3	r_3	r_3		
I_{16}	S_2			S_3		S_4		S_5			
I_{17}	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	r_2	I_{17}	
I_{18}	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6	r_6		

REDMI NOTE 8
Scanned with CamScanner