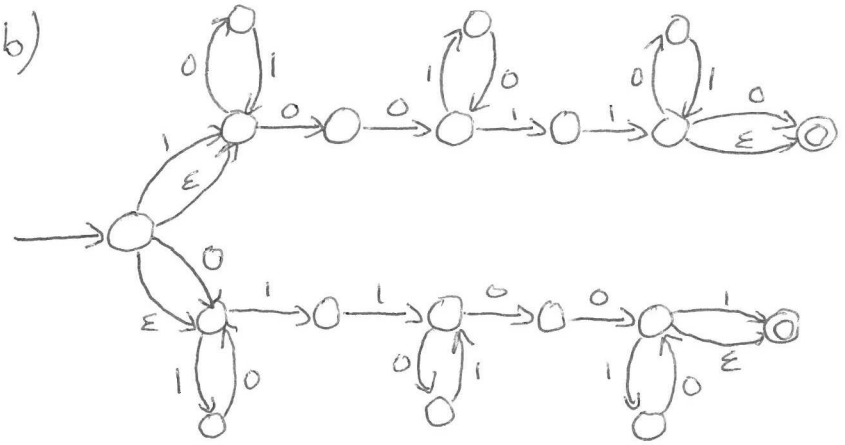
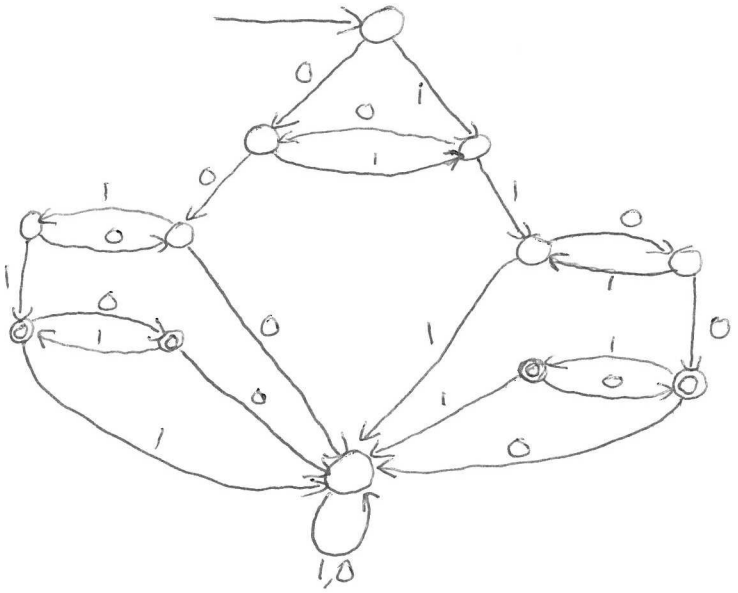
1. $\Sigma = \{0, 1\}$ $L = \{w : w \text{ contains } 00 \text{ once and } 11 \text{ once}\}$.

a) $(1|\varepsilon)(01)^*00(10)^*11(01)^*(0|\varepsilon)$
$|(0|\varepsilon)(10)^*11(01)^*00(10)^*(1|\varepsilon)$

b)



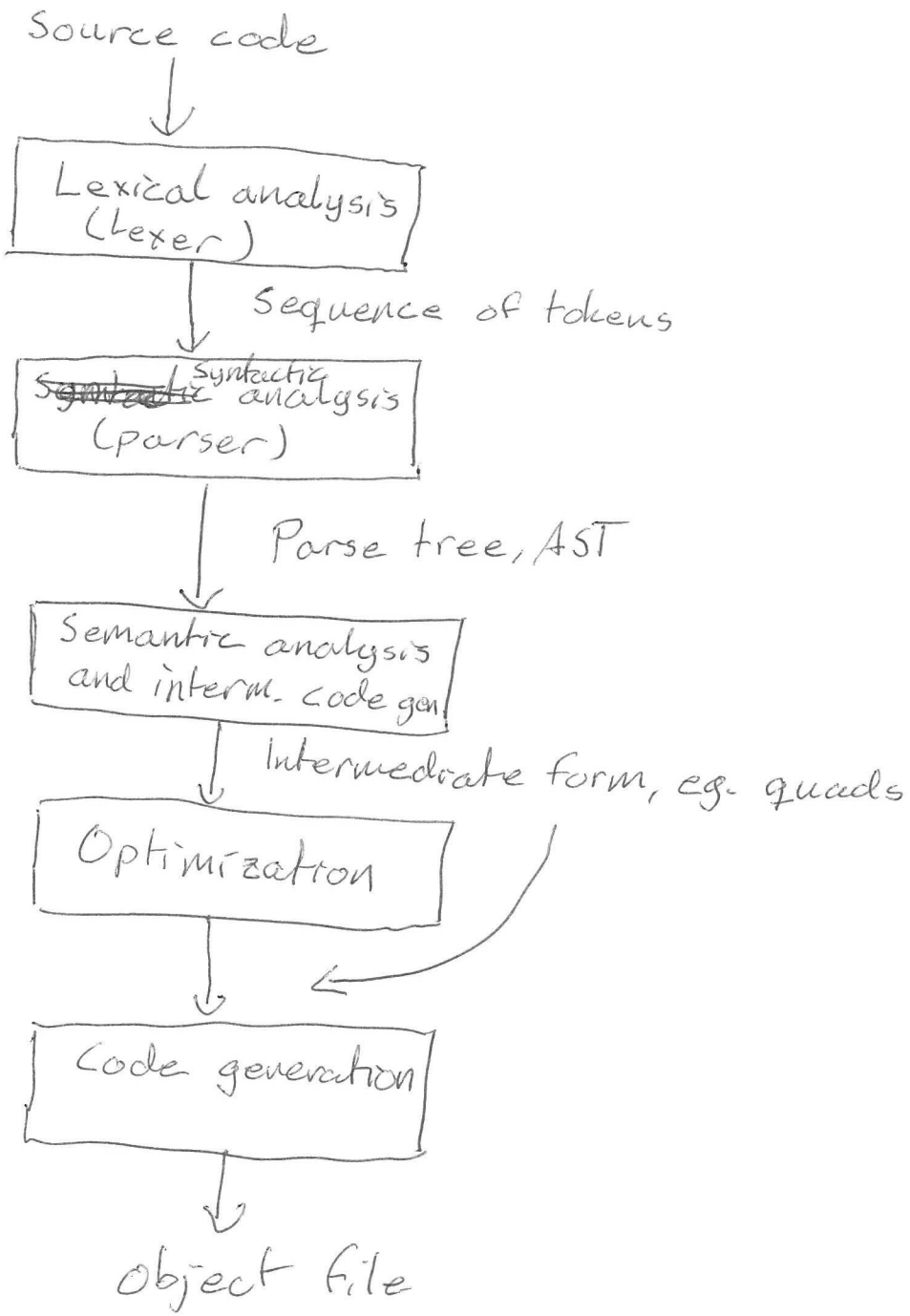
c)



2. a) Advantages: - Wider scope allows better code generation, better optimizations.
- Some languages require multi-pass.
- More modular design.

Disadvantages: - Higher compile times and memory consumption.

b)



3. a) Copy current entry in the block table one entry down, increase current block.

b) Decrease current block. For all entries in the table between the old and the new current block: Point the entries in the hash table to the hash link in that entry.

c) Find the entry in the hash table, if one exists see if it is in the current block. If not, add a new entry to the symbol table and point its hash link to the previous one. Increase the current entry in the block table.

d) Look up the variable in the hash table, follow the pointer to the symbol table.

4. a) · The grammar is Left-recursive and requires more than one token lookahead.

Grammar:

$S \to Su \mid PQv \mid PQw$

$P \to Px \mid y$

$Q \to Qz \mid \varepsilon$

P is rewritten: $A \to A\alpha \mid \beta \Rightarrow A \to \beta A'$, $A' \to \alpha A' \mid \varepsilon$

$P \to yP'$

$P' \to xP' \mid \varepsilon$

Q rewritten:

$Q \to Q'$

$Q' \to zQ' \mid \varepsilon$ —simplify→ $Q \to zQ \mid \varepsilon$

S rewritten:

$S \to Su \mid S_1$

$S_1 \to PQv \mid PQw$

$S \to S_1 S_2$

$S_2 \to uS_2 \mid \varepsilon$

$S_1 \to PQS_3$

$S_3 \to v \mid w$

Result:

$S \to S_1 S_2$

$S_1 \to PQS_3$

$S_2 \to uS_2 \mid \varepsilon$

$S_3 \to v \mid w$

$P \to yP'$

$P' \to xP' \mid \varepsilon$

$Q \to zQ \mid \varepsilon$

4 a)

```
parse() {
    S()
}

S() {
    S1();
    S2();
}

S1() {
    P();
    Q();
    S3();
}

S2() {
    if (token = u) {
        scan();
        S2();
    }
}

S3() {
    if (token ≠ v
        && token ≠ w) {
        error();
    }
    scan();
}

P() {
    if (token ≠ y)
        error();
    scan();
    P'();
}

P'() {
    if (token = x) {
        scan();
        P'();
    }
}

Q() {
    if (token = z) {
        scan();
        Q();
    }
}
```

b) Call stack.

5 a)

1. S -> P * P
2. P -> Q + P
3. | Q
4. Q -> Q - R
5. | R
6. R -> x
7. | y

	Action						Goto			
State	$	*	+	-	x	y	S	P	Q	R
00	-	-	-	-	S9	S10	1	2	5	8
01	A	-	-	-	-	-	-	-	-	-
02	-	S3	-	-	-	-	-	-	-	-
03	-	-	-	-	S9	S10	-	4	5	8
04	R1	-	-	-	-	-	-	-	-	-
05	R3	R3	S6	S11	-	-	-	-	-	-
06	-	-	-	-	S9	S10	-	7	5	8
07	R2	R2	-	-	-	-	-	-	-	-
08	R5	R5	R5	R5	-	-	-	-	-	-
09	R6	R6	R6	R6	-	-	-	-	-	-
10	R7	R7	R7	R7	-	-	-	-	-	-
11	-	-	-	-	S9	S10	-	-	-	12
12	R4	R4	R4	R4	-	-	-	-	-	-

Stack	Input
0	x+y*x-y$
0x9	+y*x-y$
0R8	+y*x-y$
0Q5	+y*x-y$
0Q5+6	y*x-y$
0Q5+6y10	*x-y$
0Q5+6R8	*x-y$
0Q5+6Q5	*x-y$
0Q5+6P7	*x-y$
0P2	*x-y$
0P2*3	x-y$
0P2*3x9	-y$
0P2*3R8	-y$
0P2*3Q5	-y$
0P2*3Q5-11	y$
0P2*3Q5-11y10	$
0P2*3Q5-11R12	$
0P2*3Q5	$
0P2*3P9	$
0S1	$

→ Accept!

b) A conflict is when we want to write two different shifts/reduces in the same cell in a table. This can be solved by rewriting the grammar or using a more powerful LR-variant.

7. a)

```
<block> ::= <block_start> <stmt_list> end
{
    pop(block_starts);
}

<block_start> ::= begin
{
    push(block_starts, quads.count);
}

<stmt> ::= restartblock
{
    gen_quad(q_jmp, 0, 0, block_starts.top);
}
```

b) We would need a structure containing all exitblock in each block so we can update the offsets in them when <block> is reduced.

8. a) LL and LR-parsers have the valid prefix property. This means that they will report an error as soon as the parsed prefix is not a valid prefix of the language.

Eg. `x = 3 then` reports an error when 'then' is parsed.

b) The parser may perform local corrections if an error is discovered.

c) The parser finds the syntax tree of the correct string with a minimum edit distance to the given, erroneous string.

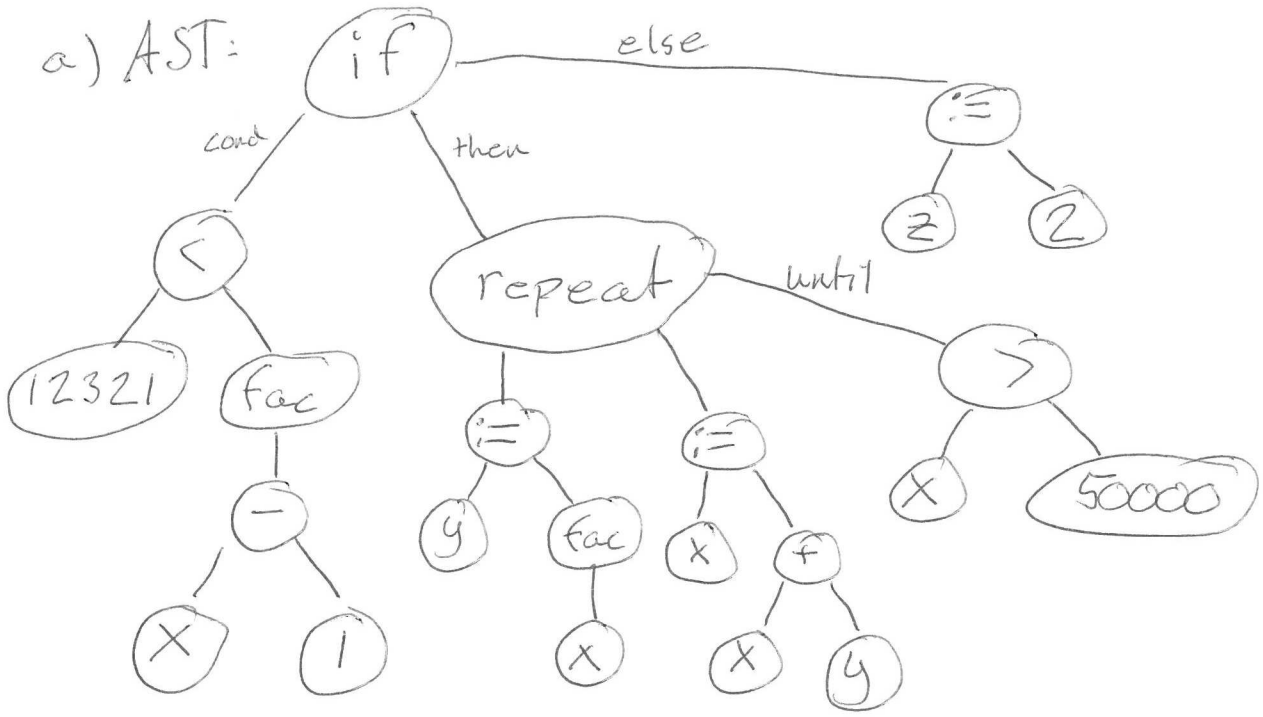
Methods that do not assume a valid prefix but assume a (mostly) valid prefix are called global correction in [ASU]. An example is minimum distance error correction (see above)

9. a) Static link is a pointer to the most recent activation of the syntactically enclosing block or function.

b) A display contains a set of static links, one for each enclosing function.

c) store the min- and max values as a header to the actual array data.

10 a) AST:



Postfix:

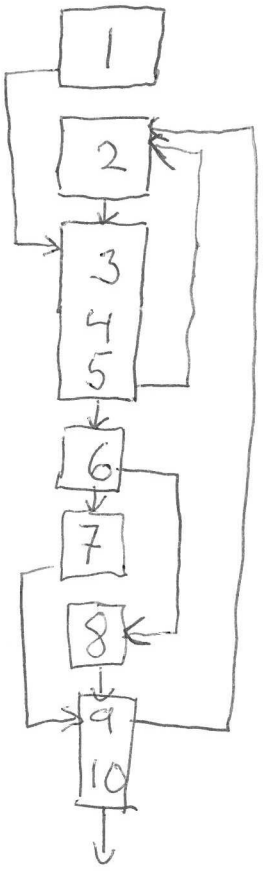
```
    12321 x 1 - fac < L1 JMPZ
L2: y x fac :=
    x x y + :=
    x 50000 > L2 JMPZ
    L3 JMP
L1: z 2 :=
L3:
```

Quads:

	load	12321	–	$1
	load	1	–	$2
	sub	X	$2	$3
	param	$3	–	–
	call	fac	1	$4
	cmp_Lt	$1	$4	$5
	jmp_F	L1	$5	–
L2:	param	X	–	–
	call	fac	1	$6
	assign	$6	–	y
	add	X	Y	$7
	assign	$7	–	X
	load	50000	–	$8
	jmp_F	L2	$9	–
	jmp	L3	–	–
L1:	load	2	–	$10
	assign	$10	–	Z
L3:				

10. b)

```
1.      goto L2
2.  L1: x := x+1
3.  L2: x := x+1
4.      x := x+1
5.      if x=1 then goto L1
6.  L3: if x=2 then goto L4
7.      goto L5
8.  L4: x := x+1
9.  L5: x := x+1
10.     if x=4 then goto L1
```



There is one loop: 2 – 10, since it is strongly connected and has one entry point.