Assignment - 3 CS335

Name: Sakshi RollNo: 180653

Q 1. Assumptions:

a) All expressions e_i have attributes truelist, falselist corresponding to incomplete goto statements in the true branch and false branch.

b) Extra attribute breaklist and continuelist have been added for S.

** Markers M_i have been added in the grammar rules.

Rules:

$S \rightarrow$ for (e_1 ; M_1 e_2 ; M_2 e_3) M_3 S_1

{ S.nextlist = merge(e2.falselist, S1.breaklist)
backpatch(S1.nextlist, M_2.quad)
backpatch(S1.continuelist, M_2.quad)
backpatch(e2.truelist, M_3.quad)
backpatch(e1.falselist, M_1.quad)
backpatch(e1.truelist, M_1.quad)

backpatch (e3. falselist , M1. quad)
backpatch (e3. truelist , M1. quad)
emit (goto M2. quad)
}

S → break
{ S. breaklist = makelist (nextinstr)
emit (goto ___)
}

S → continue
{ S. continuelist = makelist (nextinstr)
emit (goto ___)
}

S → assignment { }
// mentioned in lecture : Not supposed to have any goto call. Code will be emitted in the assignment production rule

$S \rightarrow S_1 ; M_1 S_2$
{ S. nextlist = S2. nextlist
S. breaklist = merge (S_1. breaklist , S_2. breaklist)
S. continuelist = merge (S_1. continuelist , S_2. continuelist)
backpatch (S1. nextlist , M1. quad)
}

M → ∈ { M. quad = nextinstr }

Q 2. Assumptions:

a) next is an inherited attribute with its standard meaning as in lectures.

b) default comes at the end of all cases.

c) Lnext points to after switch case stmts.

d) New attribute t is used to store E.addr.

e) We use Markess M, N for emiting "goto label" statements.

f) New attr label is used for markees.

Rules:

S → switch F { clauses }
{
clauses.t = E.addr
clauses.Lnext = S.Lnext
clauses.next = S.next
S.Lnext = S.next
}

clauses → clause ; M clauses1
{
clause.next = newlabel()
M.label = clause.next
clauses1.Lnext = clauses.Lnext
clause.Lnext = clauses.Lnext
clauses1.next = clauses.next

clauses1.t = clauses.addr

clause.t = clauses.addr

}

clauses → clause

{ clause.lnext = clauses.lnext

clause.t = clauses.t

clause.next = clauses.next

}

clause → case const : N S

{ S.next = clause.next

S.lnext = clause.lnext

N.t = clause.addr

N.label = S.next

N.val = const.val /* extra attr val for N */

}

N → ∈

{ emit (if N.addr != N.val goto N.label)

}

S → break

{ emit (goto S.lnext) }

S → others

{ others.next = S.next

others.lnext = S.lnext

}

$S \rightarrow S_1 \; ; \; M \; S_2$

{ $S_1.next = newlabel()$

$M.label = S1.next$

$S1.Lnext = S.Lnext$

$S2.Lnext = S.Lnext$

}

* Marker M is used twice i.e. in 2 productions

$M \rightarrow \epsilon$

{ emit (M.label:) }