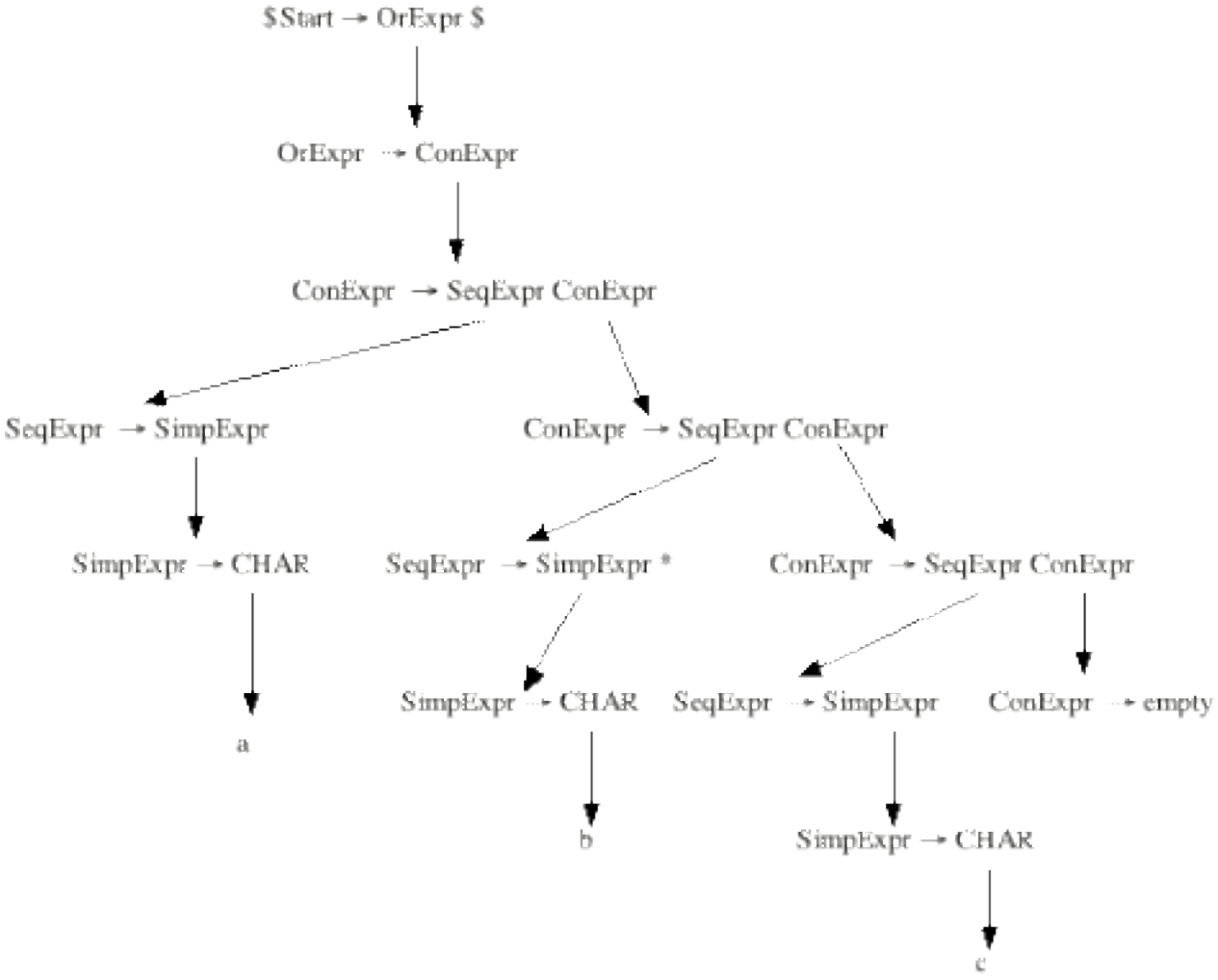
1. Bottom Up LALR(1) Parsing **[12 Marks]**

Stack					Token	Action
$ 0					CHAR a	Shift 6
$ 0	CHAR 6				CHAR b	SimpExpr → CHAR
$ 0	Simp 4					SeqExpr → SimpExpr
$ 0	Seq 2					shift CHAR 6
$ 0	Seq 2	CHAR 6			*	SimpExpr → CHAR
$ 0	Seq 2	Simp 4				Shift * 7
$ 0	Seq 2	Simp 4	* 7		CHAR c	SeqExpr → SimpExpr *
$ 0	Seq 2	Seq 2				Shift CHAR 6
$ 0	Seq 2	Seq 2	CHAR 6		$	SimpExpr → CHAR
$ 0	Seq 2	Seq 2	Simp 4			SeqExpr → SimpExpr
$ 0	Seq 2	Seq 2	Seq 2			ConExpr → empty
$ 0	Seq 2	Seq 2	Seq 2	Con 11		ConExpr → SeqExpr ConExpr
$ 0	Seq 2	Seq 2	Con 11			ConExpr → SeqExpr ConExpr
$ 0	Seq 2	Con 11				ConExpr → SeqExpr ConExpr
$ 0	Con 5					OrExpr → ConExpr
$ 0	Or 3					Shift $ 8
$ 0	Or 3	$ 8				$Start → OrExpr $
$ 0	$Start -1					Accept

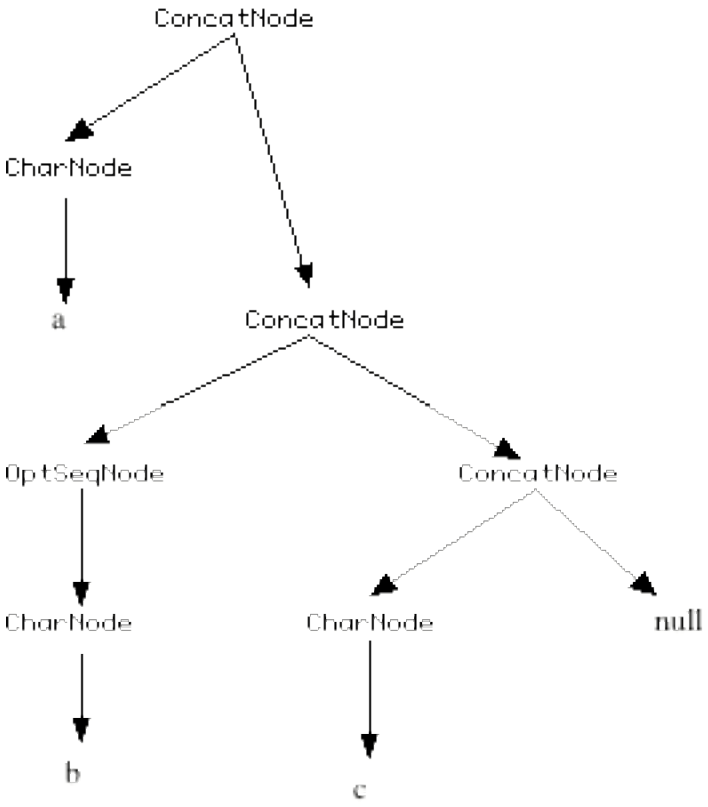
(8 marks)

(b) Indicate the parse tree corresponding to the grammar rules used to parse this input



(2 marks)

(c) Indicate the abstract syntax tree actually built by the actions in the CUP program



(2 mark

2. JLex **[6 Marks]**

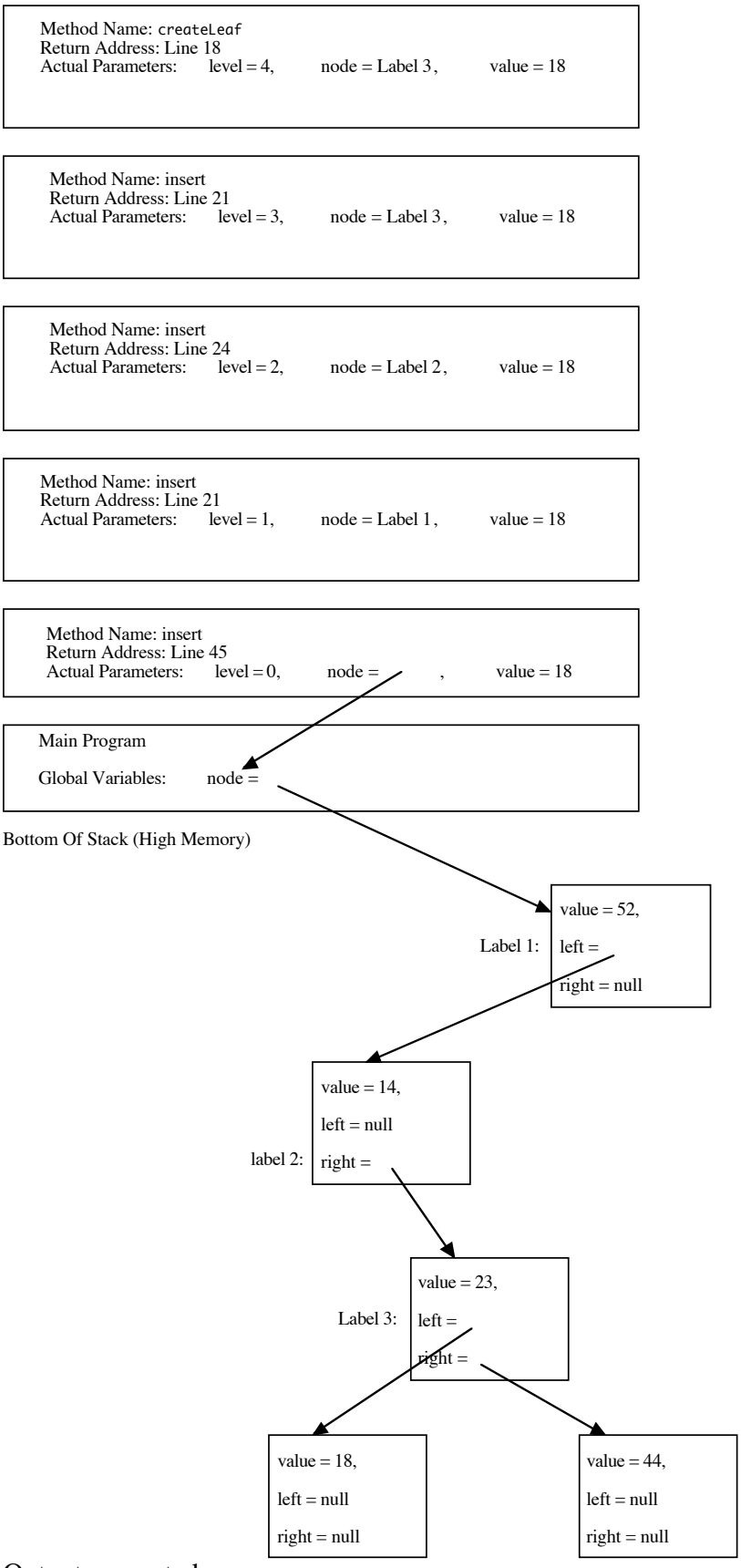
```
%{
int level = 0;
%}

%%

"{"            { level++; }
"}"            { --level; }
\"[^\"\\]*\"   { if ( level == 0 ) System.out.println( yytext() ); }
\r|\n|\r\n     { if ( level == 0 ) System.out.println( yytext() ); }
.              { if ( level == 0 ) System.out.println( yytext() ); }
```

3. Write grammar definitions [14 Marks]

```
terminal
     LEFT, RIGHT,
     LEFTSQ, RIGHTSQ,
     SEMICOLON, COMMA, ASSIGN,
     TYPE, SET, STRUCT, ENUM;

terminal String IDENT;

non terminal TypedeclarationNode TypeDeclaration;
non terminal TypeDeclListNode TypeDeclList;
non terminal TypeDeclNode TypeDecl;
non terminal TypeNode Type;
non terminal DeclStmtListNode
     FieldDeclList;
non terminal DeclStmtNode
     FieldDecl;
non terminal DeclaratorListNode
     IdentList;

start with TypeDeclaration;

TypeDeclaration::=
          TYPE TypeDeclList
     ;

TypeDeclList::=
          TypeDecl
     |
          TypeDeclList COMMA TypeDecl
     ;

TypeDecl::=
          IDENT ASSIGN Type
     ;

Type::=
          IDENT
     |
          LEFTSQ RIGHTSQ Type
     |
          STRUCT LEFT FieldDeclList RIGHT
     |
          ENUM LEFT IdentList RIGHT
     |
          SET LEFT Type RIGHT
     ;

FieldDeclList::=
          FieldDeclList SEMICOLON FieldDecl
     |
          FieldDecl
     ;

FieldDecl::=
          Type IdentList
     ;

IdentList::=
          IdentList COMMA IDENT
     |
          IDENT
     ;
```

4. Show the run time stack. [12 Marks]

Top Of Stack (Low Memory)



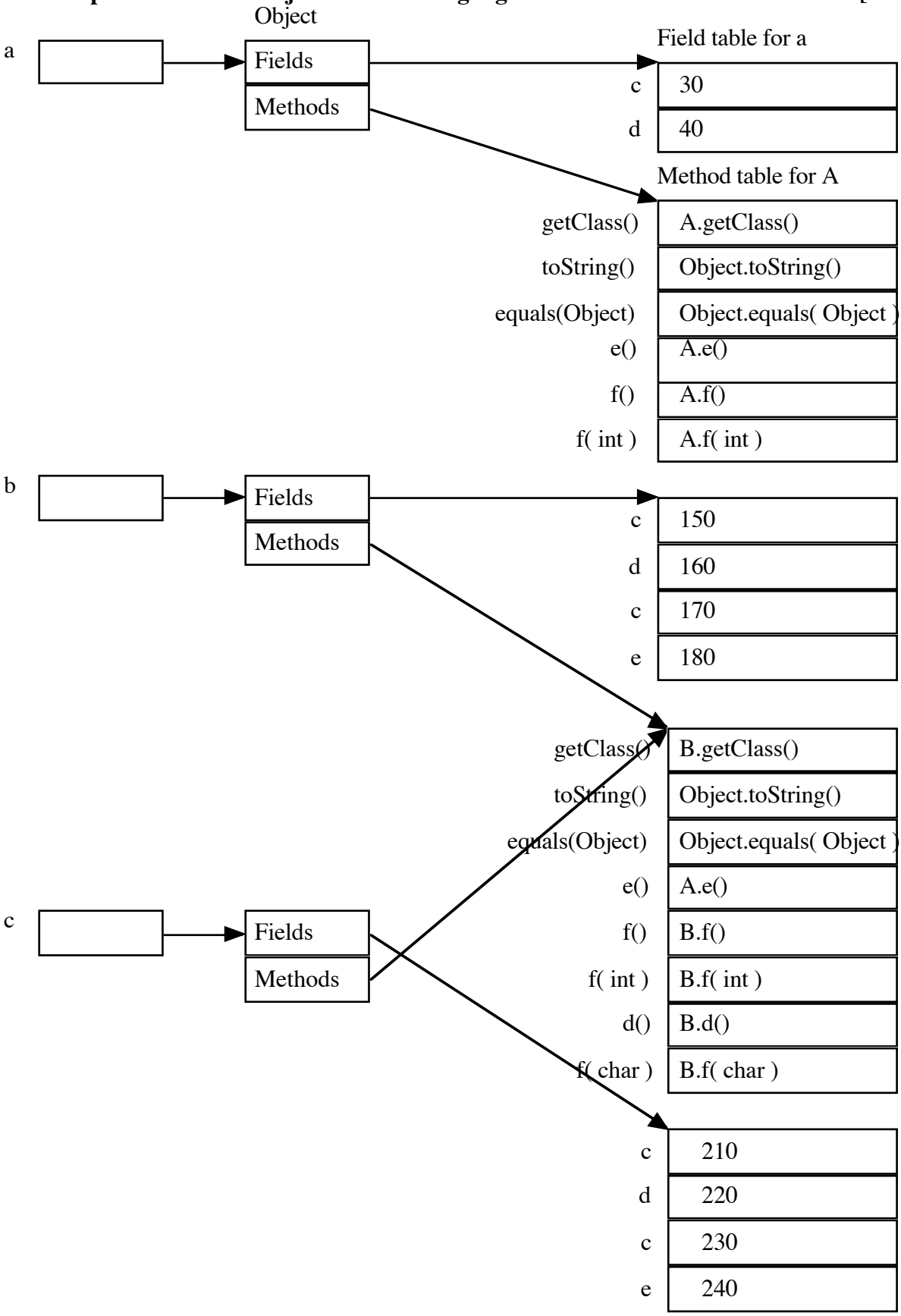
Output generated:

14 18 23 44 52

5. Type checking and Evaluation **[8 Marks]**

```
package  node.exprNode;
import  node.*;
import  env.*;
import  type.*;
import  runEnv.*;

public  class  MemberNode  extends  ExprNode  {

     private  String  typeIdent;
     private  String  memberIdent;
     private  Env  env;
     private  Decl  typeDecl;
     private  Decl  memberDecl;
     private  EnumType  enumType;

     public  MemberNode(  String  typeIdent,  String  memberIdent  )  {
          this.typeIdent  =  typeIdent;
          this.memberIdent  =  memberIdent;
          precedence  =  8;
          }

     public  String  toString()  {
          return  typeIdent  +  "."  +  memberIdent;
          }

     public  void  genEnv(  Env  env  )  {
          this.env  =  env;
          }

     public  Type  checkType()  {
          typeDecl  =  env.searchEnv(  typeIdent  );
          if  (  typeDecl  ==  null  )
               error(  "Undeclared  identifier  "  +  typeIdent  );
          if  (  typeDecl.section()  !=  Decl.SECT_TYPE  )
               error(  "Undeclared  type  identifier  "  +  typeIdent  );
          TypeType  tType  =  (  TypeType  )  typeDecl.type();
          Type  eType  =  tType.type();
          if  (  !  (  eType  instanceof  EnumType  )  )
                    error(  "attempting  to  select  non-enumerated  type  "  +  eType  );
          type  =  enumType  =  (  EnumType  )  eType;
          memberDecl  =  enumType.env().searchLocal(  memberIdent  );
          if  (  memberDecl  ==  null  )
               error(  "Undeclared  enumerated  field  "  +  memberIdent  );
          return  type;
          }

     public  RunValue  eval(  RunEnv  runEnv  )  {
          return  new  EnumValue(  enumType,  memberDecl.offset()  );
          }
     }
```

6. Implementation of object oriented languages **[10 Marks]**



```
a.c = 30
a.d = 40
b.c = 170
b.d = 160
b.e = 180
c.c = 210
c.d = 220
c.e() = A.e()
c.f() = B.f()
b.f( 'A' ) = B.f( 'A' )
c.f( 'A' ) = B.f( 65 )
c.f( 65 ) = B.f( 65 )
c.h() = A.h
```

7. Code generation **[8 Marks]**

```
lda     $sp,    -24($sp);
ldiq    $t0,    5;
stq     $t0,    0($sp);
ldiq    $t0,    6;
stq     $t0,    8($sp);
ldiq    $t0,    7;
stq     $t0,    16($sp);
ldq     $t0,    fields($ip);
ldq     $nip,   a($t0);
ldq     $t0,    methods($nip);
ldq     $pv,    f($t0);
jsr     ($pv);
lda     $sp,    +24($sp)
```