Computer Science 330 Language Implementation

Examination Information 2005

This Examination is out of 100 Marks. Attempt ALL questions. Write your answers in the spaces provided in this question and answer booklet. Do not remove the staples from the question and answer booklet. However, you may detach and remove the staples from the appendices.

Note: The examination is quite heavily related to the assignments. It is important that you do the assignments.

1		24
2		14
3		15
4		15
5		16
6		16
Total		100

1. Bottom Up LALR(1) Parsing [24 Marks]

Consider the CUP grammar: ...

(a) Using the information provided in the appendix, perform a shift-reduce LALR(1) parse of the input ...

(14 marks)

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

(4 marks)

(c) State ... is ...

Derive the set of items of State ... = GoTo(State ..., ...). Remember to take the closure to get the full set of items, and remember to compute the follow symbols.

(6 marks)

Preparation:

Assignment 1, previous tests and examinations.

2. Write a grammar definition [14 Marks]

Write a grammar definition for .... You do not have to write any actions.

Preparation:

Assignments 2, 3. Previous tests and examinations. Java grammar. Designing grammars for various constructs.

3. Interpretation [15 Marks]

Questions related to assignment 2.

Preparation:

A good (high level) understanding of Assignment 2. Chapter 8.

4. Show the run time stack. [15 Marks]

Use the program in the appendix written in the Chapter 8 INTERP7 language. Complete the drawing of the data structure built for the global variables .... Display the stack frames (activation records) for all methods in the process of being invoked when the maximum level of nesting of method invocations occurs when the statement ... is invoked, and the process is almost ready to return.

Indicate the appropriate values for each stack frame (activation record) you draw. The line numbers on the left-hand side of the program should be used to represent the return address. Draw appropriate arrows for the var parameters, and pointers to objects. For var parameters, make sure you indicate the exact field pointed to in an object very clearly.

(12 marks)

Also indicate the output generated by the complete execution of the program.

(3 marks)

Preparation:

Exercises in Chapter 8. Previous examinations. Write your own examples that involve simple data structures and recursion, with var parameters.

5. Implementation of object oriented languages [16 Marks]

Use the Java program in the appendix.

(a) Draw a diagram showing the data structures (object, field table and method table) created for the variables ... within the method Main.main. Shared data structures should be drawn only once.

(b) Indicate the output generated by the method Main.main.

Preparation:

Primarily experimentation: Write simple Java programs to test your understanding, that use overloading, overriding, constructors that explicitly or implicitly invoke the constructor of the superclass, constructors with side effects that modify static fields, accessing the same object through variables of different type, instance and static variables of the same name in different classes, implicit use of the toString method, when conversion of an object to a String, etc. Exercises in Chapter 9.

6. Code generation [16 Marks]

Use the assignment 3 OBJECT7 program in the appendix for this question.

(a) Explain the meaning or indicate or justify the legality of some statements in the language.

(b) Indicate the Alpha assembly language likely to be generated by the following statements ...

Add comments to explain the purpose of your code.

Note: An appendix is provided with common Alpha instructions.

Note: Addresses are represented using 8 bytes on the Alpha.

Preparation:

Assignment 3, especially the parts you had to implement.

福

Bruce Hutton

Exercises

1. Bottom Up LALR(1) Parsing

Consider the grammar

```
terminal IDENT, LEFT, RIGHT, COMMA, SEMICOLON, ETC, ERROR;

non terminal MethodHeader, FormalParamDeclList, FormalParamDecl, Type, IdentList;

start with MethodHeader;

MethodHeader::=
          Type IDENT LEFT FormalParamDeclList RIGHT
     ;

FormalParamDeclList::=
          FormalParamDecl
     |
          ETC
     |
          FormalParamDecl SEMICOLON FormalParamDeclList
     ;

FormalParamDecl::=
          Type IdentList
     ;

Type::=
          IDENT
     ;

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

with lexical analyser

```
ident      =    [A-Za-z][A-Za-z0-9]*
space      =    [\ \t]
newline         =    \r|\n|\r\n

%%

"("             { return token( sym.LEFT ); }
")"             { return token( sym.RIGHT ); }
","             { return token( sym.COMMA ); }
";"             { return token( sym.SEMICOLON ); }
"..."      { return token( sym.ETC ); }
{newline}       { lineNumber++; }
{space}         { }

{ident}         { return token( sym.IDENT ); }

.               { return token( sym.ERROR ); }
<<EOF>>         { return token( sym.EOF ); }
```

The action and goto tables are

-------- ACTION_TABLE --------

```
From state #0
     IDENT:SHIFT(3)
From state #1
```

```
     EOF:SHIFT(17)
From state #2
     IDENT:SHIFT(4)
From state #3
     IDENT:REDUCE(6)
From state #4
     LEFT:SHIFT(5)
From state #5
     IDENT:SHIFT(3) ETC:SHIFT(6)
From state #6
     RIGHT:REDUCE(3)
From state #7
     IDENT:SHIFT(13)
From state #8
     RIGHT:SHIFT(12)
From state #9
     RIGHT:REDUCE(2) SEMICOLON:SHIFT(10)
From state #10
     IDENT:SHIFT(3) ETC:SHIFT(6)
From state #11
     RIGHT:REDUCE(4)
From state #12
     EOF:REDUCE(1)
From state #13
     RIGHT:REDUCE(7) COMMA:REDUCE(7) SEMICOLON:REDUCE(7)
From state #14
     RIGHT:REDUCE(5) COMMA:SHIFT(15) SEMICOLON:REDUCE(5)
From state #15
     IDENT:SHIFT(16)
From state #16
     RIGHT:REDUCE(8) COMMA:REDUCE(8) SEMICOLON:REDUCE(8)
From state #17
     EOF:REDUCE(0)
```

-------- REDUCE_TABLE --------

```
From state #0:
     MethodHeader:GOTO(1)
     Type:GOTO(2)
From state #1:
From state #2:
From state #3:
From state #4:
From state #5:
     FormalParamDeclList:GOTO(8)
     FormalParamDecl:GOTO(9)
     Type:GOTO(7)
From state #6:
From state #7:
     IdentList:GOTO(14)
From state #8:
From state #9:
From state #10:
     FormalParamDeclList:GOTO(11)
     FormalParamDecl:GOTO(9)
     Type:GOTO(7)
From state #11:
From state #12:
From state #13:
From state #14:
From state #15:
From state #16:
From state #17:
```

Perform an LALR(1) parse of the input

```
int f( int a; char b, c; ... )
```

State 4 is

```
lalr_state [4]: {
  [MethodHeader ::= Type IDENT (*) LEFT FormalParamDeclList RIGHT , {EOF }]
}
transition on LEFT to state [5]
```

Derive state 5 = goto(state 4, LEFT).

2. Write a grammar definition

Grammar for a grammar.

Variable declarations.

Method declarations.

If statements with the if ... end structure of assignment 2.

Interface declarations.

Class declarations, allowing implements ...

4. Show the run time stack.

Programs/list_append/program.in

```
class List( int value;  List next; );

void printList( List a; ) {
     print( "{ " );
     while ( a != null ) {
          print( a.value );
          a = a.next;
          if ( a != null )
               print( ", " );
          }
     print( " }" );
     }

void appendList( var List a; int data; ) {
     List b;
     if ( a == null )
          a = new List { data, null };
     else {
          for ( b = a; b.next != null; b = b.next ) {
               }
          b.next = new List { data, null };
          }
     }

List createList( []int data; ) {
     List a = null;
     for ( int i = 0; i < size data; i++ )
          appendList( a, data[ i ] );
     return a;
     }

[]int a = new []int { 1, 2, 4, 8, 16, 32 };
for ( int i = 0; i < size a; i++ )
     println( a[ i ] );
List b = createList( a );
printList( b );
println( "" );
```

Programs/list_concat_copy/program.in

```
class List( int value; List next; );

void printList( List source; ) {
     print( "{ " );
     while ( source != null ) {
          print( source.value );
          source = source.next;
          if ( source != null )
               print( ", " );
          }
     println( " }" );
     }

void concatList( int level; List source1, source2; var List dest; ) {
     if ( source1 == null ) {
          dest = source2;
               // At this point
          }
     else {
          dest = new List{ source1.value, null };
          concatList( level + 1, source1.next, source2, dest.next );
               // Inside the above invocation
          }
     }

List source1, source2, a2, a4, a7, a9, dest;
a9 = new List{ 9, null };
a7 = new List{ 7, a9 };
a4 = new List{ 4, null };
a2 = new List{ 2, a4 };
source1 = a7;
source2 = a2;
concatList( 0, source1, source2, dest );  // Inside this invocation
printList( source1 );
printList( source2 );
printList( dest );
```

Programs/list_delete/program.in

```
class List ( int value; List next; );

void printList( List source; ) {
     print( "{ " );
     while ( source != null ) {
          print( source.value );
          source = source.next;
          if ( source != null )
               print( ", " );
          };
     println( " }" );
     };

void createNode( int level; var List dest; int value; List next; ) {
     dest = new List{ value, next };
     };

void deleteNode( int level; List source; var List dest; int value; ) {
     if ( source == null || value < source.value ) {
          dest = source;
          }
     else if ( value == source.value ) {
          dest = source.next;
                // Show state at this point
          }
     else {
          createNode( level + 1, dest, source.value, null );
          deleteNode( level + 1, source.next, dest.next, value );
               // Inside the above invocation
          }
     };

List source, a2, a4, a6, a8, a10, dest;
a10 = new List{ 10, null };
a8 = new List{ 8, a10 };
a6 = new List{ 6, a8 };
a4 = new List{ 4, a6 };
a2 = new List{ 2, a4 };
source = a2;
deleteNode( 0, source, dest, 6 );    // Inside this invocation
printList( source );
printList( dest );
```

Programs/list_deleteAll/program.in

```
class List( int value; List next; );

void printList( List source; ) {
     print( "{ " );
     while ( source != null ) {
          print( source.value );
          source = source.next;
          if ( source != null )
               print( ", " );
          }
     println( " }" );
     }

void deleteAll( int level; int value; List source; var List dest; ) {
     if ( source == null ) {
          dest = null;
          }
     else if ( source.value == value ) {
          deleteAll( level + 1, value, source.next, dest );
          }
     else {
          dest = new List{ source.value, null };
          deleteAll( level + 1, value, source.next, dest.next );
          }
     }

List source, a1, a2a, a4, a7, a2b, a9, dest;
a9 = new List{ 9, null };
a2b = new List{ 2, a9 };
a7 = new List{ 7, a2b };
a4 = new List{ 4, a7 };
a2a = new List{ 2, a4 };
a1 = new List{ 1, a2a };
source = a1;
deleteAll( 0, 2, source, dest );     // Inside this invocation
printList( source );
printList( dest );
```

Programs/list_head/program.in

```
class List( int value; List next; );

void printList( List source; ) {
     print( "{ " );
     while ( source != null ) {
          print( source.value );
          source = source.next;
          if ( source != null )
               print( ", " );
          }
     print( " }" );
     }

void head( int n; List source; var List dest; ) {
     print( "Enter head( " + n + ", " );
     printList( source );
     print( ", " );
     printList( dest );
     println( " )" );
     if ( n == 0 ) {
          dest = null;
               // At this point
          }
     else {
          dest = new List{ source.value, null };
          print( "dest = " );
          printList( dest );
          println( "" );
          head( n - 1, source.next, dest.next );
               // Inside the above invocation
          }
     print( "Exit head( " + n + ", " );
     printList( source );
     print( ", " );
     printList( dest );
     println( " )" );
     }

List source, a2, a4, a7, a9, dest;
a9 = new List{ 9, null };
a7 = new List{ 7, a9 };
a4 = new List{ 4, a7 };
a2 = new List{ 2, a4 };
source = a2;
head( 2, source, dest );  // Inside this invocation
printList( source );
println( "" );
printList( dest );
println( "" );
```

Programs/list_insert_copy/program.in

```
class List( int value; List next; );

void printList( List source; ) {
     print( "{ " );
     while ( source != null ) {
          print( source.value );
          source = source.next;
          if ( source != null )
               print( ", " );
          }
     println( " }" );
     }

void createNode( int level; var List dest; int value; List next; ) {
     dest = new List{ value, next };
      // Show state at this point
     }

void insertList( int level; List source; var List dest; int value; ) {
     if ( source == null || value <= source.value ) {
          createNode( level + 1, dest, value, source );
               // Inside the above invocation
          }
     else {
          createNode( level + 1, dest, source.value, null );
          insertList( level + 1, source.next, dest.next, value );
               // Inside the above invocation
          }
     }

List source, a2, a4, a7, a9, dest;
a9 = new List{ 9, null };
a7 = new List{ 7, a9 };
a4 = new List{ 4, a7 };
a2 = new List{ 2, a4 };
source = a2;
insertList( 0, source, dest, 5 );    // Inside this invocation
printList( source );
printList( dest );
```

Programs/list_insert_modify/program.in

```
class List( int value; List next; );

void printList( List a; ) {
     print( "{ " );
     while ( a != null ) {
          print( a.value );
          a = a.next;
          if ( a != null )
               print( ", " );
          }
     print( " }" );
     }

void appendHead( var List a; int data; ) {
     a = new List{ data, a };
     }

void insertList( var List a; int data; ) {
     if ( a == null || data <= a.value )
          appendHead( a, data );
     else
          insertList( a.next, data );
     }

List a5 = new List{ 5, null };
List a3 = new List{ 3, a5 };
List a1 = new List{ 1, a3 };
List a = a1;
insertList( a, 4 );
printList( a );
println( "" );
```

Programs/list_reprint1/program.in

```
class List( int value; List next; );

void appendText( var []char result; []char text; ) {
     result = result + text;
     }

void buildListText( List a; var []char result; ) {
     if ( a != null ) {
          appendText( result, a.value );
          if ( a.next != null ) {
               appendText( result, ", " );
               buildListText( a.next, result );
               }
          }
     }

List a5 = new List{ 5, null };
List a3 = new List{ 3, a5 };
List a1 = new List{ 1, a3 };
List a = a1;
[]char result = "{ ";
buildListText( a, result );
appendText( result, " }" );
println( result );
```

Programs/list_reprint2/program.in

```
class List( int value; List next; );

void buildListText( List a; var []char result; ) {
   []char result1;
     if ( a != null ) {
          if ( a.next == null )
               result = "" + a.value;
          else {
               buildListText( a.next, result1 );
               result = a.value + ", " + result1;
               }
          }
     }

List a5 = new List{ 5, null };
List a3 = new List{ 3, a5 };
List a1 = new List{ 1, a3 };
List a = a1;
[]char result;
buildListText( a, result );
println( "{" + result + "}" );
```

Programs/list_reverse/program.in

```
class List( int value; List next; );

void printList( List source; ) {
     print( "{ " );
     while ( source != null ) {
          print( source.value );
          source = source.next;
          if ( source != null )
               print( ", " );
          }
     println( " }" );
     }

void reverseTransferList( int level; List source; var List dest; ) {
     if ( source == null ) {
               // At this point
          }
     else {
          dest = new List{ source.value, dest };
          reverseTransferList( level + 1, source.next, dest );
               // Inside the above invocation
          }
     }

List source, a2, a4, a7, a9, dest;
a9 = new List{ 9, null };
a7 = new List{ 7, a9 };
a4 = new List{ 4, a7 };
a2 = new List{ 2, a4 };
source = a2;
reverseTransferList( 0, source, dest );   // Inside this invocation
printList( source );
printList( dest );
```

4b. Show the run time stack.

Make up more recursive examples, including:

Create a copy of the first n elements of a list.

For example, if source is { 2, 4, 6, 7, 9, 10, 12, 15 }

```
head( 5, source, dest )
```

might make dest { 2, 4, 6, 7, 9 }.

Return the tail of a list, with the first n elements deleted.

For example, if source is { 2, 4, 6, 7, 9, 10, 12, 15 }

```
tail( 5, source, dest )
```

might make dest { 10, 12, 15 }.

Search for a specific element of a list and create a copy of the list with that element deleted.

Create a copy of an unordered list with all elements that are < n deleted.

In all cases, make appropriate use of recursion and var parameters, and make all methods return void.

5. Implementation of object oriented languages

Draw diagrams to show the data structures generated, and output generated for

```
class A {
     public static int a = 1, b = 2;

     public int x = 5;

     public A( int x ) {
          System.out.println( "Invoke A( " + x + " )" );
          this.x = x;
          }
     public A() {
          System.out.println( "Invoke A()" );
          a++;
          x++;
          }

     public String toString() { return "A.toString(): " + x; }
     public void set( int x ) { this.x = x; }
     public int get() { return x; }
     public String f( int i ) { return "A.f( " + i + " )"; }
     public String f( double x ) { return "A.f( " + x + " )"; }
     }

class B extends A {
     public static int a = 3, d = 4;

     public int x = 100;

     public B( int x ) {
          System.out.println( "Invoke B( " + x + " )" );
          d++;
          this.x = x;
          }

     public void set( int x ) { this.x = x; }
     public int get() { return x; }
     public String f( char c ) { return "B.f( '" + c + "' )"; }
     public String f( double x ) { return "B.f( " + x + " )"; }
     }

public class Main {
     public static void main( String[] args ) {

          A a1 = new A( 55 );
          A a2 = new A();
          B b1 = new B( 66 );
          A b2 = b1;

          System.out.println( "A.a = " + A.a );
          System.out.println( "A.b = " + A.b );
          System.out.println( "B.a = " + B.a );
          System.out.println( "B.d = " + B.d );
          System.out.println();

          System.out.println( "a1.x = " + a1.x );
          System.out.println( "a2.x = " + a2.x );
          System.out.println( "b1.x = " + b1.x );
          System.out.println( "b2.x = " + b2.x );
          System.out.println();
```

```
          System.out.println( "a1 = " + a1 );
          System.out.println( "a2 = " + a2 );
          System.out.println( "b1 = " + b1 );
          System.out.println( "b2 = " + b2 );
          System.out.println();

          // 'A' is ASCII 65
          System.out.println( "a1.f( 'A' ) = " + a1.f( 'A' ) );
          System.out.println( "b1.f( 'A' ) = " + b1.f( 'A' ) );
          System.out.println( "b2.f( 'A' ) = " + b2.f( 'A' ) );
          System.out.println();

          System.out.println( "a1.f( 65 ) = " + a1.f( 65 ) );
          System.out.println( "b1.f( 65 ) = " + b1.f( 65 ) );
          System.out.println( "b2.f( 65 ) = " + b2.f( 65 ) );
          System.out.println();

          System.out.println( "a1.f( 65.0 ) = " + a1.f( 65.0 ) );
          System.out.println( "b1.f( 65.0 ) = " + b1.f( 65.0 ) );
          System.out.println( "b2.f( 65.0 ) = " + b2.f( 65.0 ) );
          System.out.println();

          b1.set( 123 );
          b2.set( 456 );
          System.out.println( "b1.x = " + b1.x );
          System.out.println( "b2.x = " + b2.x );
          System.out.println( "b1.get() = " + b1.get() );
          System.out.println( "b2.get() = " + b2.get() );
          }
     }
```

What is the output:

```
class A {
     int initA() {
          System.out.println( "Init A fields" );
          return 0;
          }
     int x = initA();
     A() {
          System.out.println( "Init A default constructor" );
          }
     }

class B extends A {
     int initB() {
          System.out.println( "Init B fields" );
          return 0;
          }
     int x = initB();
     B() {
          System.out.println( "Init B default constructor" );
          }
     }

class Main {
     public static void main( String[] arg ) {
          B b = new B();
          }
     }
```