CS 442/542 Final Exam
Due 11:59 PM Thursday May 13
Final Exam
Question 1 (15 Points)

Build the LL(1) parse table for the following grammar.

1. S -> X
2. X -> A
3. X -> L
4. A -> i
5. A -> d
6. L -> ( R )
7. R -> X Y
8. Y -> R
9. Y -> ε
Final Exam

Question 2 (15 Points)

Build the LR(1) action and goto tables for the following grammar.

1. $S \rightarrow A$
2. $A \rightarrow ( A )$
3. $A \rightarrow a$
Final Exam
Question 3 (30 Points)

• Implement an **interpreter** for the language defined by the grammar shown on slide 4.

• Programs in the language print the contents of a list.

• The values in a list are integers.

• The integers are either explicitly listed or they are the results of evaluation of an addition or multiplication function.

• The input program comes from stdin
Final Exam
Question 3 Example Program

• The following shows an example program.

Print(2,3,4);

Print(+(2,3),(4,6));

Print(+(+(4,5,*(2,3,2))),99,*(2,2,2), *(+(1,2,3,4),*(2,5)));

• The output of the program is
  2 3 4
  5 24
  21 99 16 100
Final Exam Question 3 Grammar
The productions are numbered so they can be referred to on the next slide

1. Prog -> StmtSeq
2. StmtSeq -> Stmt StmtSeq
3. StmtSeq -> ε
4. Stmt -> Print ( List ) ;
5. List -> List , Item
6. List -> Item
7. Item -> Func ( List )
8. Item -> IntLit
9. Func -> +
10. Func -> *
Final Exam Question 3
Action Hints

• There are no actions to take for productions 1, 2 and 3. These productions exist so a program can have multiple print statements.

• The action for production 4 is to print the values in the list

• The actions for productions 5 and 6 build a list

• The action for production 7 evaluates the function (either + or *). This evaluation produces an integer (i.e. the data type for Item is int)

• IntLit is an integer literal (a sequence of 1 or more digits)
Final Exam
Question 4 (40 Points)

• Implement an compiler for the language defined by the grammar shown on slide 8.

• Since this is a compiler you will generate MIPS code like you did for the project.

• The language is a simple string processing language.

• The language includes features to declare a string with a maximum size, initialize a string, store the result of concatenating 2 strings into another string.

• Like you did in the project your input will come from IOMngr.
null[1];
init(null, "");
init(x, "abc");
init(y,"def");
concat(z,x,y);
concat(w, z, y);
print z;
print w;
concat(w, x, null);
print w;
concat(w, x, x);
print w;

The output of the program is

abcdef
abcdefdef
abc
abcabc
Final Exam Question 4 Grammar
The productions are numbered so they can be referred to on the next slide

1. Prog -> Declarations StmtSeq
2. Declarations -> Dec Declarations
3. Declarations -> ε
4. Dec -> Id [ IntLit ] ;
5. StmtSeq -> Stmt StmtSeq
6. StmtSeq -> ε
7. Stmt -> concat(Id, Id, Id) ;
8. Stmt -> init(Id, Str);
9. Stmt -> print Id
10. Id -> Ident
11. Str -> StrLit
Final Exam Question 4
Action Hints

• The action for production 1 should be a call to a finish function similar to your semester project.

• There are no actions to take for productions 2 and 3. These productions exist so a program can have multiple declarations.

• The action for production 4 is entering information into a symbol table.

• The actions for productions 5 and 6 will be similar to those you used in the semester project (i.e. build the list of instructions associated with the statements).

• The action for production 7 is to store in the memory associated with the first Id the concatenation of the current values associated with the second and third Id (see the example output on a previous slide). The first Id must be different than the second and third Ids. You can assume this is the case. You do not have to check for this error.

• The action for production 8 is to store a copy of the value associated with Str to the memory location associated with Id.

• IntLit is an integer literal (a sequence of 1 or more digits).

• StrList is a string literal (A “ followed by a sequence of 0 or more uppercase or lowercase letters followed by a “). You will have to create an entry in the data section of your MIPs program for each string literal.

An example entry is L3: .asciiz "def"
Final Exam Question 4 MIPS Hint.
The MIPS code shown below is a subroutine that can copy a string. The source address is passed in register $a0 and the destination address is passed in $a1. When the subroutine returns, $v0 has the address of the null character that ends the new string. Note in a .data section line like L3: .asciiz "def" The “def” is terminated with a null character.

```
strCopy:
    move  $t0, $a0
    move  $t1, $a1
loop:
    lb    $t2, 0($t0)
    beq   $t2, $zero, end
    sb    $t2, 0($t1)
    addi  $t0, $t0, 1
    addi  $t1, $t1, 1
    j     loop
end:
    move  $v0, $t1
    sb    $zero, 0($t1)
    jr     $ra
```
Final Exam Submission

• Upload to Canvas one zip file. The zip file must contain a pdf, a folder for question 3 and a folder for question 4. The pdf must contain your answers to questions 1 and 2. The folder for question 3 must contain files final3.l (the lex file), final4.y (the yacc file; this file must contain the semantic routines and a main function), a sample input file and a file containing the output of your program when the sample input was used. The folder for question 4 must contain files final4.l (the lex file), final4.y (the yacc file), final4Sem.h, final4Sem.c (the semantic files), final4Main.c, a sample input file, the MIPS code generated by the sample input file and a file containing the output of your MIPS program when the sample input was used. I will use my own SymTab, IOMngr, CodeGen

• You can assume the input for questions 3 and 4 are syntactically correct.