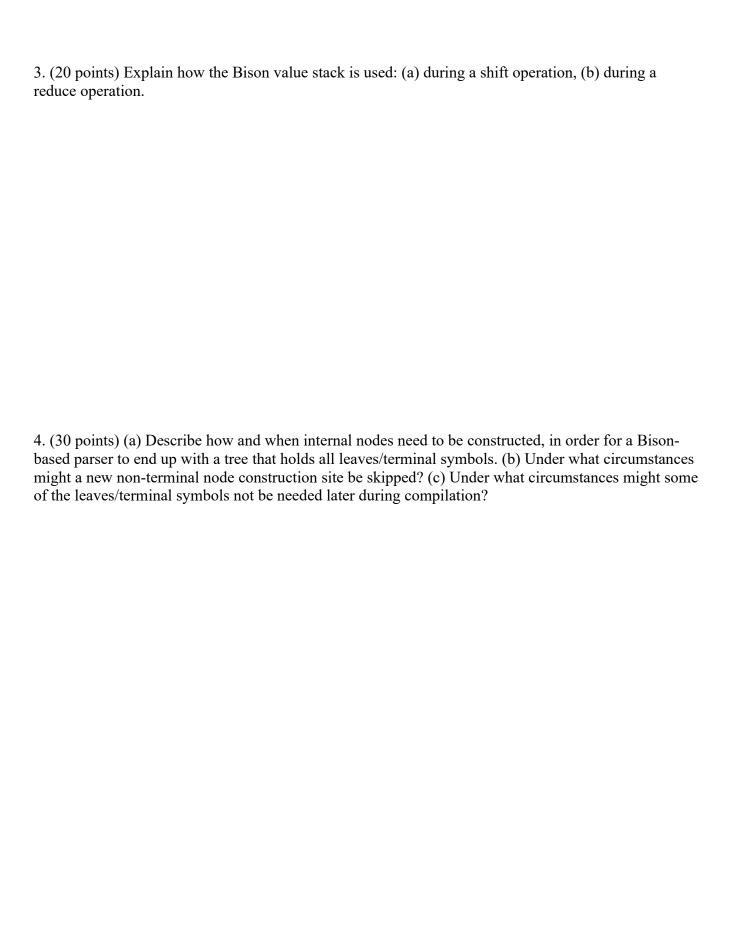
CS 423 Midterm Examination	Name:	
	_	

1. (20 points) Explain why the comments in most programming languages are generally discarded by the lexical analyzer, rather than being returned as a token with an integer category like all the other parts of the source code.

2. (30 points) (a) Write a **regular expression** (you may use Flex extended regular expression operators) for the set of strings described by the context free grammar below. You may use {id} to denote usual regular expression for C/C++ variable names for IDENT, namely [a-zA-Z_][a-zA-Z0-9]*. (b) Under what circumstances is it better to use regular expressions, and under what circumstances is it better to use context free grammars?

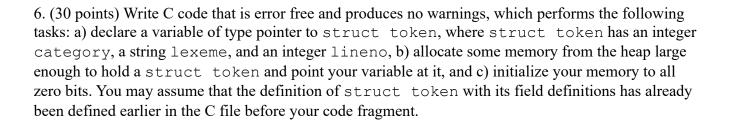
```
declaration : type_specifier decl_list ';' ;
type_specifier : INT | CHAR | DOUBLE ;
decl_list : decl | decl ',' decl_list ;
decl: IDENT | '*' IDENT | IDENT '[' INTCONST ']' ;
```



5. (40 points) Consider the following grammar for C struct declarations, given in YACC-style syntax.

```
vd : cl t dl ';' | cl t ';';
cl : STATIC | DYNAMIC | /* epsilon */;
t : INT | STRUCT '{' vl '}';
dl : d | d ',' dl;
d : IDENT | '*' d | d '[' INTCONST ']';
vl : vl vd | /* epsilon */;
```

a) What are the terminal symbols? b) What are the nonterminal symbols? c) Which nonterminals have recursive productions? d) Why might some recursive grammar rules use epsilon, while others do not? e) Rewrite this grammar so that it recognizes the same declarations but does not use epsilon production rules, if that is possible. If not, explain why not.



7. (20 points) In looking at yydebug output, you might notice that it appeared like the same terminal symbol (for example, a semi-colon) was repeated over and over again in the output, even through sections of parsing where no syntax error occurred. Why might the same terminal symbol appear on the input repeatedly through several iterations of a shift-reduce parser?

