System Programming-for VIVA

1. yylval: It is a global variable shared between the lexical analyzer and parser

2. yytext: a buffer that holds the input characters that actually match the pattern (i.e lexeme) or say a pointer to the matched string .

3. yyleng: length of the lexeme found

4. How the lexical analyzer solve conflict in input?
Sol.

- Always prefer a longer prefix to a shorter prefix.
- If the longest possible prefix matches two or more patterns, prefer the pattern listed first in the Lex program.

5. YACC(yet another complier complier) based on the concept of LALR parser

6. lex and yacc program communicate with each other using lex.yy.c and y.tab.h header files respectively

7. To generate a y.tab.h header file we need to complier our program using –d option i.e. on terminal yacc –d filename.y

8. Yacc uses two rules resolve conflicts (default rules)

- A reduce/reduce conflict is resolved by choosing the conflicting production listed first in the Yacc specification.
- A shift/reduce conflict is resolved in favor of shift. This rule resolves the shift/reduce conflict arising from the dangling-else ambiguity correctly.

9. REJECT: It means continue to the next expression that matches the current input and print the result according to whatever rule was the second best choice.

10. yyin :- the input stream pointer (i.e it points to an input file which is to be scanned or tokenised), however the default input of default main() is stdin .

11. yylex() :- implies the main entry point for lex, reads the input stream generates tokens, returns zero at the end of input stream . It is called to invoke the lexer (or scanner) and each time yylex() is called, the scanner continues processing the input from where it last left off.

12. yywrap() :- it is called by lex when input is exhausted (or at EOF). default yywrap always return 1.
13. yyparse() :- it parses (i.e builds the parse tree) of lexeme.

14. YYERROR : Cause an immediate syntax error. This statement initiates error recovery just as if the parser itself had detected an error; however, it does not call yyerror, and does not print any message. If you want to print an error message, call yyerror explicitly before the `YYERROR;' statement.

Q. what is lex.yy.c file? What does y.tab.h and y.tab.c do?

NOTE: In lab systems--lex package name is flex and yacc package name is bison. Both are open source software.

Q1. What are the different phases of compiler?
Sol. In the order-

- Lexical analyzer,
- Syntax analyzer,
- Semantic analyzer,
- Intermediate code generator,
- Machine independent code optimizer,
- Code generator,
- Machine-dependent code optimizer

Q2. Tokens, pattern and lexeme
Sol. **Tokens:** a pair consisting of token name and an optional attributes value. The token names are the input symbols that the parser processes.

Pattern: A pattern is a description of the form that the lexemes of a token may take. In the case of a keyword as a token, the pattern is just the sequence of characters that form the keyword.

Lexeme: A lexeme is a sequence of characters in the source program that matches the pattern for a token and is identified by the lexical analyzer as an instance of that token.

Q3. Error recovery actions in lexical analyzer
Sol.

- Delete one character from the remaining input.
- Insert a missing character into the remaining input.
- Replace a character by another character.
- Transpose two adjacent characters.

Q4. Define Handle.
Sol. a handle of a right-sentential form γ (gamma) is a production $A \rightarrow \beta$ and a position of γ where β may be found, such that replacing, β at that position by A produces the previous right-sentential form in a rightmost derivation of γ

Q5. Types of Conflict in parsers

- Shift/reduce conflict: whether to shift new symbol onto the stack or perform a reduction
- Reduce/reduce conflict: cannot decide which of several reduction to make

Q6. What does the L,R and k stands for in LR(k) parser?
Sol. L-left to right and R is rightmost derivation. K- number of input symbols of lookahead
When k is omitted ,it is assumed to be 1.

Q7. Kernel items and non-kernel items
Sol.

- Kernel items: the initial item, S'→. S, and all items whose dots are not at the left end.
- Non-kernel items: all items with their dots at the left end, except for S' → .S

Q8. Difference between SLR and LR(0) parsing table
Sol. In SLR table reduction will be performed according to the FOLLOW of left hand side i.e. one lookahead is present
While in LR table reduction will be performed in a whole row of terminals present in the grammar

Q9. Define Viable prefix
Sol. A viable prefix is a prefix of a right-sentential form that does not continue past the right end of the rightmost handle of that sentential form.

10. If there are no shift reduce/reduce reduce conflict present in CLR(1),then there will be no shift reduce conflict in LALR(1) but it could have a reduce reduce conflict.

11. Relationship between number of states in LR(0), SLR,LALR and CLR
LR(0) = SLR = LALR <= CLR

12. Least powerful parser-LR(0)
Most powerful parser-CLR(1)

13. LR(0), SLR,LALR and CLR can parse only unambiguous grammar and Operator precedence parser can parse some of the ambiguous grammar (using associative and precedence of operator)

14. Error recovery in LR parsers

- Panic mode recovery
- Phrase level recovery

15. Two kinds of attributes for non terminals:

- Synthesized attribute: The attribute which takes data values from its child nodes, is called synthesized attribute. These are also called s-attributed production.

- Inherited attribute: The attribute which takes values form parents or sibling nodes are called inherited attributes. The production rule having inherited attribute are called L-attributed productions.

16. Define dependency graph
Sol. A dependency graph depicts the flow of information among the attribute instances in a particular parse tree; an edge from one attribute instance to another means that the value of the first is needed to compute the second.
17. **Three address code**: at most one operator on the right side of an instruction
Quadruples: A quadruple has four fields, which we call op, arg,, arg2, and result.
Triples: A triple has only three fields, which we call op, arg,, and arg2.
Indirect triples: Indirect triples consist of a listing of pointers to triples, rather than a listing of triples themselves.

Q18. Define Activation Tree
Sol. The activations of procedures during the running of an entire program by a tree, called an activation tree. Each node corresponds to one activation, and the root is the activation of the main procedure that initiates execution of the program.

Q19. Content of activation record
Sol.
- Temporary values
- Local data
- Saved machine status
- Access link
- Control link
- Returned values
- Actual parameters

20. Calling sequence and division of tasks between caller and callee
i. The caller evaluates the actual parameters.
ii. The caller stores a return address and the old value of *top-sp* into the callee's activation record.
iii. The callee saves the register values and other status information.
iv. The callee initializes its local data and begins execution.

21. Return sequence is:
i. The callee places the return value next to the parameters
ii. Using information in the machine-status field, the callee restores top-sp and other registers, and then branches to the return address that the caller placed in the status field.
iii. The caller knows where the return value is, relative to the current value of top-sp.

Q22. What are the major data structures used in the assembly process?
- Machine Opcode Table (MOT): It holds the opcodes used by the processor for different instructions mnemonics

- Pseudo Opcode Table (POT): Pseudo opcodes supported by the assembler.eg: DB(define byte),DW(define word),RESB(reserve byte)
- Symbol Table

Q23. Define Symbol Table
Sol. It is a data structure used by the assembler to remember information about the identifiers appearing in the program to be assembled. The types of symbols that are stored in the symbol table include variables, procedures, functions, labels.
Q24.Two pass assembler does not suffer from the problem of forward referencing .why?
Sol. The entire pass I is dedicated to find the symbols defined in the input program. At the end of pass I information about all the symbols are available in the symbol table. Pass II simply uses theses values o generate the code.

Q25. What does Linking and Loader do?
Sol. Linking is the process of combining different object modules into one executable file. Loader loads the program into physical memory

Q26. Difference between static libraries and shared libraries

Q27.DLL Hell: If the shared library is deleted, moved or renamed or if an incompatible version of the DLL is copied to a place that is earlier in the search, the executable could malfunction or even fail to load. On windows this is called DLL Hell