**VIVA QUESTIONS**

**Define system software**.

System software is computer software designed to operate the computer hardware and to provide a platform for running application software. Eg: operating system, assembler, and loader.

**What is an Assembler?**

Assembler for an assembly language, a computer program to translate between lower-level representations of computer programs.

**Explain lex and yacc tools**

Lex: - scanner that can identify those tokens

Yacc: - parser.yacc takes a concise description of a grammar and produces a C routine that can parse that grammar.

**Explain yyleng?**

yyleng-contains the length of the string our lexer recognizes.

**What is a Parser?**

A Parser for a Grammar is a program which takes in the Language string as it's input and produces either a corresponding Parse tree or an Error.

**What is the Syntax of a Language?**

The Rules which tells whether a string is a valid Program or not are called the Syntax.

**What is the Semantics of a Language?**

The Rules which gives meaning to programs are called the Semantics of a Language.

**What are tokens?**

When a string representing a program is broken into sequence of substrings, such that each substring represents a constant, identifier, operator, keyword etc of the language, these substrings are called the tokens of the Language.

**What is the Lexical Analysis?**

The Function of a lexical Analyzer is to read the input stream representing the Source program, one character at a time and to translate it into valid tokens.

**How can we represent a token in a language?**

The Tokens in a Language are represented by a set of Regular Expressions. A regular expression specifies a set of strings to be matched. It contains text characters and operator characters. The Advantage of using regular expression is that a recognizer can be automatically generated.

**How are the tokens recognized?**

The tokens which are represented by an Regular Expressions are recognized in an input string by means of a state transition Diagram and Finite Automata.

**Are Lexical Analysis and Parsing two different Passes?**

These two can form two different passes of a Parser. The Lexical analysis can store all the recognized tokens in an intermediate file and give it to the Parser as an input. However it is more convenient to have the lexical Analyzer as a co routine or a subroutine which the Parser calls whenever it requires a token.

**What are the Advantages of using Context-Free**

**grammars?** It is precise and easy to understand.

It is easier to determine syntactic ambiguities and conflicts in the grammar.

**If Context-free grammars can represent every regular expression, why do one needs R.E at all?**

Regular Expression are Simpler than Context-free grammars.

**Dept. of CSE, MSEC Page 49**

System Software and Operating System Lab-17CSL67 VI Semester CSE

It is easier to construct a recognizer for R.E than Context-Free grammar.

Breaking the Syntactic structure into Lexical & non-Lexical parts provide better front end for the Parser.

R.E are most powerful in describing the lexical constructs like identifiers, keywords etc while Context-free grammars in representing the nested or block structures of the Language.

**What are the Parse Trees?**

Parse trees are the Graphical representation of the grammar which filters out the choice for replacement order of the Production rules.

**What are Terminals and non-Terminals in a grammar?**

Terminals:- All the basic symbols or tokens of which the language is composed of are called Terminals. In a Parse Tree the Leafs represents the Terminal Symbol. Non-Terminals:- These are syntactic variables in the grammar which represents a set of strings the grammar is composed of. In a Parse tree all the inner nodes represents the Non-Terminal symbols.

**What are Ambiguous Grammars?**

A Grammar that produces more than one Parse Tree for the same sentences or the Production rules in a grammar is said to be ambiguous.

**What is bottom up Parsing?**

The Parsing method is which the Parse tree is constructed from the input language string beginning from the leaves and going up to the root node.

Bottom-Up parsing is also called shift-reduce parsing due to its implementation. The YACC supports shift-reduce parsing.

**What is the need of Operator precedence?**

The shift reduce Parsing has a basic limitation. Grammars which can represent a left-sentential parse tree as well as right-sentential parse tree cannot be handled by shift reduce parsing. Such a grammar ought to have two non-terminals in the production rule. So the Terminal sandwiched between these two non-terminals must have some associability and precedence. This will help the parser to understand which non-terminal would be expanded first.

**What is exit status command?**

Exit 0- return success, command executed successfully.

Exit 1 – return failure.

**Define API’s**

An application programming interface (API) is a source code based specification intended to be used as an interface by software components to communicate with each other.

**1. What does the ‘Definition section’ contain?**

Ans: Substitutions code and start states. This section will be copied into lex.yy.c.

**2. What does the ‘Rule section’ contain?**

Ans: Defines how to scan and what action to take for ach token

**3. What does the last section contain?**

Ans: C auxiliary subroutines: any user code and scanning function yylex()

**4. What is the difference between yylex() and scanf().**

Ans: yylex() is used to accept input and call parser, but scanf() for only accepting data.

**5. How is the instruction in the first section used?**

Ans: It is copied as it is into lex.yy.c

**6. Explain the procedure for executing a lex prg.**

Ans: The prg is written in ‘.l’ or ‘.lex’ extension eg.”abc.lex”

This is first lexxed as ‘ lex abc.lex –l’ here –l means link with lex library

This provides a ‘c’ program called ‘lex.yy.c’

Now this ‘c’ prg is compiled using ‘cc’ command and run as ./a.out

**7. What is the rule of lex to specify file?**

Ans: <pattern> {corresponding action}

**8. Explain longest match**

Ans: the action for the longest matching pattern is execution

**9. Explain more applicable rule.**

Ans: the action for the most applicable matching pattern is execution

Ex. If input is “post”

Rule is: rule1 = “post” {print(“hi”);}

Rule2= “[a-zA-Z]+ {print(“hello”)

The rule1 is applied

**10. What does the lex prg contain?**

Ans: A specification of a lexical analyzer is prepared by creating a program lex.l in

the lex language. Then this lex is run thru the lex compiler to produce a c prglex.

yy.c

**11. what are the different parts of lex prg?**

Ans: Declarations, Translation rules and auxiliary section

**12. what do the following meta symbols indicate?**

Ans: matches any single char except the \n

\* matches zero or more copies of preceding expression.

+ matches one or more occurrence of the preceding regular expression

eg.: [0-9]+ matches 1 or 1111 or 12345 but not empty string.

? matches zero or occurrence of preceding regular expression.

Eg. : ?[0-9]+ matches a signed number with or without minus sign.

| is like ‘OR’

**13. What is a parser?**

Ans: parse generator can be used to facilitate the construction of the front end of a

compiler.

**14. is Yacc a compler!**

Ans: No, Yacc is available as a command on the unix system and has been used to

implement of hundred of compiler.

**15. Explain construction of yacc program.**

Ans: a program containing yacc specification is provided to yacc. This provides

y.tab.c as a c prg. This is compiled using c compiler to get exe ie a.out.

**16. Different sections of yacc**

Ans: Declarations, translation rule and support & routine sections.

**17. provide the grammar of yacc**

Ans: <left side> : <alternate1> {semantic action1}

| alternate2> {semantic action2}

**18. what is the difference between Lex and Yacc?**

Ans: lex prg to lex.yy.c to scanner to parser to exe

yac prg to y.tab.c to parser to exe

**19. what is the treatment of lines written between first %{ and }% ?**

Ans: This part will be embedded into x.tab.c

**20.Define system software**.

System software is computer software designed to operate the computer hardware and

to provide a platform for running application software. Eg: operating system,

assembler, and loader.

**21.What is an Assembler?**

Assembler for an assembly language, a computer program to translate between lowerlevel

representations of computer programs.

**22. Explain lex and yacc tools**

 Lex: - scanner that can identify those tokens

 Yacc: - parser.yacc takes a concise description of a grammar and produces a C

routine that can parse that grammar.

**23 .Explain yyleng?**

Yyleng-contains the length of the string our lexer recognizes.

**24. What is a Parser?**

A Parser for a Grammar is a program which takes in the Language string as it's input

and produces either a corresponding Parse tree or an Error.

**25. What is the Syntax of a Language?**

The Rules which tells whether a string is a valid Program or not are called the Syntax.

**26.What is the Semantics of a Language?**

The Rules which gives meaning to programs are called the Semantics of a Language.

**27.What are tokens?**

When a string representing a program is broken into sequence of substrings, such that

each substring represents a constant, identifier, operator, keyword etc of the language,

these substrings are called the tokens of the Language.

**28.What is the Lexical Analysis?**

The Function of a lexical Analyzer is to read the input stream representing the Source

program, one character at a time and to translate it into valid tokens.

**29.How can we represent a token in a language?**

The Tokens in a Language are represented by a set of Regular Expressions. A regular

expression specifies a set of strings to be matched. It contains text characters and

operator characters. The Advantage of using regular expression is that a recognizer can

be automatically generated.

**30.How are the tokens recognized?**

The tokens which are represented by an Regular Expressions are recognized in an

input string by means of a state transition Diagram and Finite Automata.

**Operating Systems**

**1. List out the types in mainframe systems**

Ans: i) Batch system

ii) Multiprogrammed systems

iii) Time-sharing system

**2. What is file-server systems?**

Ans: File-server system provides a file system interface where clients can create, update,

read, and delete files

**3. What is job scheduling?**

Ans: If several jobs are ready to be brought in to memory, and if there is not enough room

for all of them, then the system must choose among them. Making this decision is job

scheduling.

**4. Define non-preemptive scheduling**

Ans: First come first serve scheduling (fcfs) Shortest job first scheduling

**5. Define dispatcher and its functions?**

Ans: i. Switching context

ii.Switching to user mode

iii. Jumping to the proper location in the user program to restarts the program

**6. Define dispatch latency?**

Ans: The time taken for the dispatcher to stop one process and start another running process

is known as dispatch latency.

7. **What is mean by FCFS scheduling?**

Ans: FCFS also known as First-in-first-out(FIFO)which is the simplest scheduling policy.

arriving jobs are inserted into the tail(rear)of the ready queue.

**8. Define CPU scheduling.**

Ans: CPU scheduling is the process of switching the CPU among various processes. CPU

scheduling is the basis of multi programmed operating systems. By switching the CPU

among processes, the operating system can make the computer more productive.

**9. What is a Dispatcher?**

Ans: The dispatcher is the module that gives control of the CPU to the process selected by

the short-term scheduler. This function involves: • Switching context • Switching to

user mode • Jumping to the proper location in the user program to restart that program.

**10. What is turnaround time?**

Ans: Turnaround time is the interval from the time of submission to the time of completion

of

a process. It is the sum of the periods spent waiting to get into memory, waiting in the

ready queue, executing on the CPU, and doing I/O.

**11. What are privileged instructions?**

Ans: Some of the machine instructions that may cause harm to a system are designated as

privileged instructions. The hardware allows the privileged instructions to be executed

only in monitor mode.

**12. How can a user program disrupt the normal operations of a system?**

Ans: user program may disrupt the normal operation of a system by

 Issuing illegal I/O operations

 by accessing memory locations within the OS itself

 Refusing to relinquish the CPU

**13. How is the protection for memory provided?**

Ans: The protection against illegal memory access is done by using two registers. The base

register and the limit register. The base register holds the smallest legal physical

address; the limit register contains the size of the range. The base and limit registers

can be loaded only by the OS using special privileged instructions.

**14. Define race condition.**

Ans: When several process access and manipulate same data concurrently, then the outcome

of the execution depends on particular order in which the access takes place is called

race condition. To avoid race condition, only one process at a time can manipulate the

shared variable.

**15. Define entry section and exit section.**

Ans: The critical section problem is to design a protocol that the processes can use to

Co-operate. Each process must request permission to enter its critical section. The

section of the code implementing this request is the entry section. The critical section is

followed by an exit section. The remaining code is the remainder section