UNIT 1

INTRODUCTION TO COMPILERS& LEXICAL ANALYSIS

2 marks

1. **What is a Complier?**

A Complier is a program that reads a program written in one language-the source language-and translates it in to an equivalent program in another language-the target language .The compiler reports to its user the presence of errors in the source program.

2. **What are the main two parts of compilation? What are they performing?**

The two main parts are

• Analysis part breaks up the source program into constituent pieces and creates an intermediate representation of the source program.

• Synthesis part constructs the desired target program from the intermediate representation.

3. **How many phases does analysis consists?**

Analysis consists of three phases

i .Linear analysis
ii. Hierarchical analysis
iii. Semantic analysis

4. **What is a Symbol table?**

A Symbol table is a data structure containing a record for each identifier, with fields for the attributes of the identifier. The data structure allows us to find the record for each identifier quickly and to store or retrieve data from that record quickly.

5. **State the general phases of a compiler**

i. Lexical analysis
ii. Syntax analysis
iii. Semantic analysis
iv. Intermediate code generation
v. Code optimization
vi. Code generation

6. **What is the need for separating the analysis phase into lexical analysis and parsing? (Or) What are the issues of lexical analyzer?**

• Simpler design is perhaps the most important consideration. The separation of lexical analysis from syntax analysis often allows us to simplify one or the other of these phases.

• Compiler efficiency is improved.

• Compiler portability is enhanced.

7. **What is Lexical Analysis?**

The first phase of compiler is Lexical Analysis. This is also known as linear analysis in which the stream of characters making up the source program is read from left-toright and grouped into tokens that are sequences of characters having a collective meaning.

8. What are the functions performed in analysis phase?

- Lexical analysis or Linear analysis
- Syntax analysis or hierarchical analysis
- Semantic analysis

9. What are the functions performed in synthesis phase?

- Intermediate code generation
- Code generation
- Code optimization

10. Define patterns/lexeme/tokens

A set of strings in the input for which the same token is produced as output. This set of strings described by a rule called pattern associated with the token.

A lexeme is a sequence of characters in the source program that is matched by the pattern for a token.

Token is a sequence of character that can be treated as a single logical entity.

11. What are the operations on language?

Union
Concatenation
Kleene closure or star closure and
Star closure.

12. Give the error recovery actions in lexical errors?

- Deleting an extraneous character
- Inserting a missing character
- Replacing an incorrect character by a correct character.

13. What are the three parts of lexical program?

Declarations

%%

Translation rules

%%

Auxiliary procedures

14. Write short notes on symbol table manager?

The table management or bookkeeping portion of the compiler keeps track of the names used by program and records essential information about each, such as its type (int, real etc.,) the data structure used to record this information is called a symbol table manger.

15. Write short notes on error handler

The error handler is invoked when a flaw in the source program is detected. It must warn the programmer by issuing a diagnostic, and adjust the information being passed from phase to phase so that each phase can proceed. So that as many errors as possible can be detected in one compilation.

16. What is meant by lexical analysis?

It reads the characters in the program and groups them into tokens that are sequences of characters having a collective meaning Such as an identifier, a keyword, a punctuation, character or a multi-character operator like ++.

17. What is meant by syntax analysis?

It processes the string of descriptors, synthesized by the lexical analyzer, to determine the syntactic structure of an input statement. This process is known as parsing. Output of the parsing step is a representation of the syntactic structure of a statement. It is represented in the form of syntax tree.

18. What is meant by intermediate code generation?

After syntax and semantic analysis, some compilers generate an explicit intermediate representation of the source program. It can have a variety of forms. This form called three address code. It consists of sequence of instructions, each of which has at most three operands.

19. What is meant by semantic analysis?

This phase checks the source program for semantic errors and gathers type of information for the subsequent phase.

20. What is a sentinel? What is its usage?

A Sentinel is a special character that cannot be part of the source program. Normally we use 'eof' as the sentinel. This is used for speeding-up the lexical analyzer.

21. **What are the Error-recovery actions in a lexical analyzer?**

1. Deleting an extraneous character
2. Inserting a missing character
3. Replacing an incorrect character by a correct character
4. Transposing two adjacent characters

22. What is recognizer?

Recognizers are machines. These are the machines which accept the strings belonging to certain language. If the valid strings of such language are accepted by the machine then it is said that the corresponding language is accepted by that machine, otherwise it is rejected.

23. Define Token.

The token can be defined as a meaningful group of characters over the character set of the programming language like identifiers, keywords, constants and others.

24. What do you mean by Cross-Compiler?

A cross compiler is a compiler capable of creating executable code for a platform other than the one on which the compiler is run. (ie). A compiler may run on one machine and produce target code for another machine.

PART B

1. Describe the various phases of compiler with suitable example
2. Analyze structure of compiler with an assignment statement
3. Discuss in detail about the role of Lexical analyzer with the possible error recovery actions.
4. Describe in detail about issues in lexical analysis.
5. Describe the Input buffering techniques in detail.

6. Discuss how a finite automaton is used to represent tokens and perform lexical analysis with examples.

7. Summarize in detail about how the tokens are specified by the compiler with suitable example.

8. Define Finite Automata. Differentiate Deterministic Finite Automata and Non-Deterministic Finite Automata with example

9. Solve the given regular expression (a/b)* abb (a/b)* into NFA using Thompson construction..

10. Create DFA the following NFA.

M=({q0,q1},{0,1},δ,q0,{q1})

Where δ(q0,0)={q0,q1}

δ (q0,1)={q1}

δ(q1,0)=ϕ

δ(q1,1)={q0,q1}

11.Show how the DFA is directly converted from an augmented regular expression

((ε / a) b *) * .

12.Draw NFA for the regular expression ab*/ab

13.Define the language accepted by FA. Convert the following NFA

into DFA.

	0	1
p	{p,q}	{p}
q	{r}	{r}
r	{s}	ϕ
*s	{s}	ϕ

14 Define Lex and Lex specifications. How lexical analyzer is constructed using lex? Give an example.

15 Explain an algorithm for Lex that recognizes the tokens.Describe in detail the tool for generating lexical

analyzer.

16 Analyze the algorithm for minimizing the number of states of a DFA.

17. Minimize DFA using Thompson Construction. (a / b) * a (a / b) (a / b)

17 Show the minimized DFA for the regular expression: (0 + 1) * (0 + 1) 1 0.

(a|b)*a(a|b)(a|b)(a|b).

UNIT II

SYNTAX ANALYSIS

2 marks

1. **What is the output of syntax analysis phase? What are the three general typesof parsers for grammars?**
   Parser (or) parse tree is the output of syntax analysis phase.
   General types of parsers:
   1) Universal parsing
   2) Top-down
   3) Bottom-up

2. **What are the different strategies that a parser can employ to recover from a syntactic error?**
   Panic modePhrase
   level

   Error productions

   Global correction

3. **What are the goals of error handler in a parser?**
   The error handler in a parser has simple-to-state goals:
   It should report the presence of errors clearly and accurately.
   It should recover from each error quickly enough to be able to detect subsequent errors.
   It should not significantly slow down the processing of correct programs.

4. **What is phrase level error recovery?**
   On discovering an error, a parser may perform local correction on the remaining input; that is, it may replace a prefix of the remaining input by some string that allows theparser to continue. This is known as phrase level error recovery.

5. **How will you define a context free grammar?**
   A context free grammar consists of terminals, non-terminals, a start symbol, and productions.
   i. Terminals are the basic symbols from which strings are formed. "Token" isa synonym for terminal. Ex: **if, then, else.**
   ii. Nonterminals are syntactic variables that denote sets of strings, which help define the language generated by the grammar. Ex: stmt, expr.
   iii. Start symbol is one of the nonterminals in a grammar and the set of strings it denotes is the language defined by the grammar. Ex: S.
   iv. The productions of a grammar specify the manner in which the terminals and nonterminals can be combined to form strings Ex: expr $\longrightarrow$ **id**

6. **Define context free language. When will you say that two CFGs are equal?**
   A language that can be generated by a grammar is said to be a context free language. Iftwo grammars generate the same language, the grammars are said to be equivalent.

7. **Give the definition for leftmost and canonical derivations.**
   Derivations in which only the leftmost nonterminal in any sentential form is

replaced at each step are termed leftmost derivations
Derivations in which the rightmost nonterminal is replaced at each step are termed canonical derivations.

8. What is a parse tree?

A parse tree may be viewed as a graphical representation for a derivation that filters out the choice regarding replacement order. Each interior node of a parse tree is labeled by some nonterminal A and that the children of the node are labeled from left to right by symbols in the right side of the production by which this A was replaced in the derivation. The leaves of the parse tree are terminal symbols.

9. What is an ambiguous grammar? Give an example.

A grammar that produces more than one parse tree for some sentence is said to be ambiguous
An ambiguous grammar is one that produces more than one leftmost or rightmost derivation for the same sentence.

Ex:

$E \rightarrow E+E \ / \ E*E \ / \ \mathbf{id}$

10. Why do we use regular expressions to define the lexical syntax of a language?

i. The lexical rules of a language are frequently quite simple, and to describe them wedo not need a notation as powerful as grammars.
ii. Regular expressions generally provide a more concise and easier to understand notation for tokens than grammars.
iii. More efficient lexical analyzers can be constructed automatically from regular expressions than from arbitrary grammars.
iv. Separating the syntactic structure of a language into lexical and non lexical parts provides a convenient way of modularizing the front end of a compiler into two manageable-sized components.

11. When will you call a grammar as the left recursive one?

A grammar is a left recursive if it has a nonterminal **A** such that there is a derivation A A for some string .

12. Define left factoring.

Left factoring is a grammar transformation that is useful for producing a grammar suitable for predictive parsing. The basic idea is that when it is not clear which of two alternative productions to use to expand a nonterminal "**A**", we may be able to rewrite the"**A**" productions to refer the decision until we have seen enough of the input to make the right choice.

13. Left factor the following grammar:

$S \rightarrow iEtS \mid iEtSeS \mid a$
$E \rightarrow b.$

Ans:

The left factored grammar is,S
$\rightarrow iEtSS' \mid a$
$S' \rightarrow eS \mid \varepsilon E$
$\rightarrow b$

14. What is parsing?

Parsing is the process of determining if a string of tokens can be generated by a grammar.

15. What is Top Down parsing?

Starting with the root, labeled, does the top- down construction of a parse tree with the starting nonterminal, repeatedly performing the following steps.

i. At node n, labeled with non terminal "**A**", select one of the productions for "**A**" and construct children at n for the symbols on the right side of the production.
ii. Find the next node at which a sub tree is to be constructed.

16. What do you mean by Recursive Descent Parsing?

Recursive Descent Parsing is top down method of syntax analysis in which we execute a set of recursive procedures to process the input. A procedure is associated with each nonterminal of a grammar.

17. What is meant by Predictive parsing?

A special form of Recursive Descent parsing, in which the look-ahead symbol unambiguously determines the procedure selected for each nonterminal, where no backtracking is required.

18. Define Bottom Up Parsing.

Parsing method in which construction starts at the leaves and proceeds towardsthe root is called as Bottom Up Parsing.

19. What is Shift-Reduce parsing?

A general style of bottom-up syntax analysis, which attempts to construct a parse tree for an input string beginning at the leaves and working up towards the root.

20. Define handle. What do you mean by handle pruning?

An Handle of a string is a sub string that matches the right side of production andwhose reduction to the nonterminal on the left side of the production represents one step along the reverse of a rightmost derivation.

The process of obtaining rightmost derivation in reverse is known as Handle Pruning.

21. Define LR (0) items.

An LR (0) item of a grammar G is a production of G with a dot at some positionof the right side. Thus the production A → XYZ yields the following four items,

A → .XYZA
→ X.YZA →
XY.Z A →
XYZ.

22. What do you mean by viable prefixes?

The set of prefixes of right sentential forms that can appear on the stack of a shift-reduce parser are called viable prefixes.
A viable prefix is that it is a prefix of a right sentential form that does not continue the past the right end of the rightmost handle of that sentential form.

23. What is meant by an operator grammar? Give an example.

A grammar is operator grammar if,

i. No production rule involves " " on the right side.

ii. No production has two adjacent nonterminals on the right side..

Ex:

E → E+E | E-E | E*E | E/E | E E | (E) | -E | **id**

24. What are the disadvantages of operator precedence parsing? May/June 2007

i. It is hard to handle tokens like the minus sign, which has two different precedences.

ii. Since the relationship between a grammar for the language being parsed and the operator – precedence parser itself is tenuous, one cannot always be sure the parser accepts exactly the desired language.

iii. Only a small class of grammars can be parsed using operator precedence techniques.

25. State error recovery in operator-Precedence Parsing.

There are two points in the parsing process at which an operator-precedence parser can discover the syntactic errors:

i. If no precedence relation holds between the terminal on top of the stack and the current input.

ii. If a handle has been found, but there is no production with this handle as a right side.

26. LR (k) parsing stands for what?

The "L" is for left-to-right scanning of the input, the "R" for constructing a rightmost derivation in reverse, and the k for the number of input symbols of lookahead that are used in making parsing decisions.

27. Define LR grammar.

A grammar for which we can construct a parsing table is said to be an LR grammar.

28. What are kernel and non kernel items?

i. The set of items which include the initial item, S S, and all items whose dots are not at the left end are known as kernel items.

ii. The set of items, which have their dots at the left end, are known as non kernel items.

29. Why SLR and LALR are more economical to construct than canonical LR?

For a comparison of parser size, the SLR and LALR tables for a grammar always have the same number of states, and this number is typically several hundred states for a language like Pascal. The canonical LR table would typically have several thousand states for the same size language. Thus, it is much easier and more economical toconstruct SLR and LALR tables than the canonical LR tables.

30. What is ambiguous grammar? Give an example.

A grammar G is said to be ambiguous if it generates more than one parse trees for sentence of language L(G).

Example: E-> E+E|E*E|id

PART B

1. Explain left recursion and Left Factoring.

2. Eliminate left recursion and left factoring for the following grammar.
E → E + T | E - T | T
T → a | b | (E).

3. What is an ambiguous and un ambiguous grammar?
Identify the following grammar is ambiguous or not.
E→E+E | E*E | (E)|-E |id for the sentence id+id*id

4. Illustrate the predictive parser for the following grammar.
S→ (L) | a
L→ L, S | S

5. Evaluate predictive parsing table and parse the string id+id*id. find FIRST and FOLLOW.
E→E+T | T T→T*F | F
F→(E) | id

6. Construct Stack implementation of shift reduce parsing for the grammar
E->E+E
E->E*E
E->(E)
E->id and the input string id1+id2*id3

7. Describe on detail about the role of parser.

8. Discuss about the context-free grammar.

9. What are the Error recovery techniques used in Predictive parsing? Explain in detail.

10. Analyze the following grammar is a LR(1) grammar and construct LALR parsing table.
S → Aa | bAc | dC | bda
A→ d.
Parse the input string bdc using the table generated.

11. Define SLR (1) parser. Describe the Steps for the SLR parser.

12. Predict the following grammar for generate the SLR parsing table.
E→E+T | T
T→T*F | F

13. Consider the following grammar
S → AS|b
A→SA|a.
Construct the SLR parse table for the grammar.
Show the actions of the parser for the input string "abab".

14. Examine the following grammar using canonical parsing table.

E → E + T F → (E)

E → T F → id.

T → T * F

UNIT III

SYNTAX DIRECTED TRANSLATION &INTERMEDIATE CODE GENERATION

1.What is syntax directed translation?

A syntax directed definition specifies the values of attributes by associating semantic rules with the grammar productions

Production E->E1+T

Semantic Rule E.code=E1.code||T.code||'+'

2. What is synthesized attributes?

A synthesized attribute at node N is defined only in terms of attribute values of children

of N and at N

3. What is inherited attributes ?

An inherited attribute at node N is defined only in terms of attribute values at N's parent, N itself and N's siblings

4. List the three kinds of intermediate representation.

The three kinds of intermediate representations are

i. Syntax trees
ii. Postfix notation
iii. Three address code

5. What is postfix notation?

A Postfix notation is a linearized representation of a syntax tree. It is a list ofnodes of the tree in which a node appears immediately after its children.

6. What is the usage of syntax directed definition.

Syntax trees for assignment statement are produced by the syntax directeddefinition.

7. Why "Three address code" is named so?

The reason for the term "Three address code" is that each usually containsthree addresses, two for operands and one for the result.

8. Define three-address code.

Three-address code is a sequence of statements of the general formx **:=** y *op* z

where x, y and z are names, constants, or compiler-generated temporaries; op stands for any operator, such as fixed or floating-point arithmetic operator, or a logical operator on boolean-valued data.

Three-address code is a linearized representation of a syntax tree or a dag in which explicit names correspond to the interior nodes of the graph.

9. State quadruple

A quadruple is a record structure with four fields, which we call op, arg1, arg2and result.

10. What is called an abstract or syntax tree?

A tree in which each leaf represents an operand and each interior node an operator is called as abstract or syntax tree.

11. **What are the functions used to create the nodes of syntax trees?**

Mknode (op, left, right) Mkleaf (id,entry)
Mkleaf (num, val)

12. What are the advantages of generating an intermediate representation?

Ease of conversion from the source program to the intermediate code.Ease with which subsequent processing can be performed from the intermediate code.

13. Define annotated parse tree?

A parse tree showing the values of attributes at each node is called an annotated parse tree. The process of computing an attribute values at the nodes is called annotating parse tree.

14. What are triples?

The fields arg1,and arg2 for the arguments of op, are either pointers to the symbol table or pointers into the triple structure then the three fields used in theintermediate code format are called triples.

In other words the intermediate code format is known as triples.

15. Write a short note on declarations?

Declarations in a procedure, for each local name, we create a symbol table entry with information like the type and the relative address of the storage for the name. The relative address consists of an offset from the base of the static data area or the field for local data in an activation record. The procedure enter (name, type, offset) create a symbol table entry.

16. List the types of three address statements.

The types of three address statements are

a. Assignment statements
b. Assignment Instructions
c. Copy statements
d. Unconditional Jumps
e. Conditional jumps
f. Indexed assignments
g. Address and pointer assignments
h. Procedure calls and return

17. What are the various methods of implementing three-address statements?

a. Quadruples
b. Triples
c. Indirect triples

18. What is S-Attributed Syntax Directed Translation(SDT)?

If an SDT uses only synthesized attributes, it is called as S-attributed SDT. S-attributed SDTs are evaluated in bottom-up parsing, as the values of the parent nodes depend upon the values of the child nodes.

19. What is L-Attributed Syntax Directed Translation(SDT)?

If an SDT uses either synthesized attributes or inherited attributes with a restriction that it can inherit values from left siblings only, it is called as L-attributed SDT. Attributes in L-attributed SDTs are evaluated by depth- first and left-to-right parsing manner.

20. Define Boolean Expression.

Expressions which are composed of the Boolean operators (and, or, and not) applied to elements that are Boolean variables or relational expressions are known as Boolean expressions

21. What are the two methods to represent the value of a Boolean expression?

a. The first method is to encode true and false numerically and to evaluate aBoolean expression analogously to an arithmetic expression.
b. The second principal method of implementing Boolean expression is by flow of control that is representing the value of a Boolean expression by a position reached in a program.

22. What do you mean by viable prefixes.

Viable prefixes are the set of prefixes of right sentinels forms that can appear on the stack of shift/reduce parser are called viable prefixes. It is always possible to add terminal symbols to the end of the viable prefix to obtain a right sentential form.

23. What is meant by Shot-Circuit or jumping code?

We can also translate a Boolean expression into three-address code without generating code for any of the Boolean operators and without having the code necessarily evaluate the entire expression. This style of evaluation is sometimes called “short-circuit” or “jumping” code.

24. What is known as calling sequence?

A sequence of actions taken on entry to and exit from each procedure is known ascalling sequence.

25. Define an attribute. Give the types of an attribute?

An attribute may represent any quantity, with each grammar symbol, it associates a set of attributes and with each production, a set of semantic rules for computing values of the attributes associated with the symbols appearing in that production. Example: a type, a value, a memory location etc., i) Synthesized attributes. ii) Inherited attributes.

PART B

1. Discuss the following in detail about the Syntax Directed Definitions.
(i)Inherited Atrributes and Synthesized attributes.
(ii) Evaluate SDD of a parse tree.
2. Explain the steps for constructing a DAG. Construct the DAG for the following expression
((x+y)-((x+y)*(x-y)))+((x+y)*(x-y))
3. What is Type conversion? What are the two types of type conversion? Evaluate the rules for the type conversion.
4. Generate an intermediate code for the following code segment with the required syntax-directed translation scheme.
if (a > b)
x = a + b else
x = a - b
5. Create the following for the arithmetic expression a+- (b+c)* into
(i)Syntax tree (ii)Quadruples (iii)Triples (iv)Indirect Triples
6. Describe in detail about addressing array Elements.
7. Discuss in detail about Translation of array reference.
8. Describe in detail about types and declaration with suitable example
9. Discuss in detail about
(i)Dependency graph (ii)Ordering Evaluation of Attributes

UNIT IV
RUN-TIME ENVIRONMENT AND CODE GENERATION

1. **Define an attribute. Give the types of an attribute?**

An attribute may represent any quantity, with each grammar symbol, it associates a set of attributes and with each production, a set of semantic rules for computing values of the attributes associated with the symbols appearing in that production. Example: a type, a value, a memory location etc., i) Synthesized attributes. ii) Inherited attributes.

2. **Mention the two rules for type checking.**

Type checker for a language is based on information about the syntactic constructs in the language, the notion of types, and the rules for assigning types to language constructs.

3. **When does dangling references occur?**

When there is a reference to storage that has been de-allocated, logical error occurs as it uses dangling reference where the value of de-allocated storage is undefined according to the semantics of most languages

4. **What is syntax directed translation?**

A syntax directed definition specifies the values of attributes by associating semantic rules with the grammar productions
Production E->E1+T
Semantic Rule E.code=E1.code||T.code||'+'

5. **What is synthesized attributes?**

A synthesized attribute at node N is defined only in terms of attribute values of children of N and at N

6. **What is inherited attributes ?**

An inherited attribute at node N is defined only in terms of attribute values at N's parent, N itself and N's siblings

7. **Define DAG.**

A DAG for a basic block is a directed acyclic graph with the following labels on nodes:
i) Leaves are labeled by unique identifiers, either variable names or constants.
ii) Interior nodes are labeled by an operator symbol.
iii)Nodes are also optionally given a sequence of identifiers for labels.

8. **What are the functions and properties of Memory Manager?**

Two basic functions: Allocation Deallocation
Properties of memory managers: Space efficiency
Program efficiency
Low overhead

9. What is static checking?

A compiler must check that the source program follows both syntactic and semantic conversions of the source language. This checking called static checking detects and reports programming errors.

10. Give some examples of static checking?

Type checks:
A compiler should report an error if an operator is applied to an incompatible operand.
Flow of control checks:
Statements that cause flow of control to leave a construct must have some place to which to transfer the flow of control.

11. What is a Procedure?

A procedure definition is a declaration that associates an identifier with a statement.
The identifier is the procedure name , and the statement is the procedure body.

12. What is the use of a control stack?

A control stack is used to keep track of live procedure activations. The idea is to push the node for an activation onto the control stack as the activation begins and to pop the node when the activation ends.

13. What are the types of storage allocation strategies? (OR) List Dynamic Storage allocation techniques.

Static allocation : Lays out storage for all objects at compile time.
Stack allocation : Manages the run-time storage as a stack.
Heap allocation : Allocates and deallocates storage as needed at run time from a data area known as heap

14. **Define dependency graph**

If an attribute b at a node in a parse tree depends on an attribute c, then the semantic rule for b at the node must be evaluated after the semantic rule that defines c. The interdependencies among the inherited and synthesized attributes at the nodes in a parse tree can be depicted by a directed graph called dependency graph.

15. What are the functions used to create the nodes of syntax tree?

mknode(op,left,right) mkleaf(id, entry)
mkleaf(num,val)

16. What is a type expression?

The type of a language construct will be denoted by a "type expression" .
Informally a type expression is either a basic type or is formed by applying an operator called a type constructor to other type expressions.

17. What is a type system?

A type system is a collection of rules for assigning type expressions to the various parts of a program. A type checker implements a type system.

18. What are coercions?
Conversion from one type to another is said to be implicit if it is to be done automatically by the compiler. Implicit type conversions are also called coercions.

19. What is an intermediate code?
Intermediate codes are machine independent codes, but they are close to machine instructions. The given program in a source language is converted to an equivalent program in an intermediate languaue by the intermediate code generator.

20. What are quadruples?
Quadruples are close to machine instructions, but they are nor actual machine instructions.

21. What is three address code?
We use the term “three address code” because each statement usually contains three addresses (two for operands, one for the result).
General form : X:=Y op Z

22. What are the representations of three address code?
Quadruples
Triples
Indirect triples.

UNIT IV

PART B

1. Compare static versus dynamic memory allocation.
2. Explain in detail about the various issues in code generation with examples
3. Discuss in detail about the activation tree and activation record with suitable example
4. Explain in detail about instruction selection and register allocation of code generation.
5. Illustrate in detail about the code generation algorithm with an example.
6. Describe the usage of stack in the memory allocation and discuss in detail about stack allocation space of memory.
7. Define the heap management of memory and describe in detail
8. Compare the stack and heap allocation memory in detail with suitable examples
9. Create following assignment statement into three address code
   D:=(a-b)*(a-c)+(a-c)
   Apply code generation algorithm to generate a code sequence for the three address statement.

10. Generate code for the following sequence assuming that n is in a

    memory location

```
 s=0

i=0

L1 : if I > n goto L2
      s=s+i

      i=i+1
      goto L1

      L2:
```

UNIT V
CODE OPTIMIZATION

1. **What are basic blocks?**

A sequence of consecutive statements which may be entered only at the beginning and when entered are executed in sequence without halt or possibility of branch, are called basic blocks

2. **What do you mean by copy propagation?**

After the assignment of one variable to another, a reference to one variable may be replaced with the value of the other variable.
If w := x appears in a block, all subsequent uses of w can be replaced with uses of x.

Before	After
b := z + y	b := z + y
a := b	a := b
x := 2 * b	x := 2 * a

3. **What is a flow graph?**

The basic block and their successor relationships shown by a directed graph is called a flow graph. The nodes of a flow graph are the basic blocks.

4. **Write the three address code sequence for the assignment statement**.

d:=(a-b)+(a-c)+(a-c) t1=a-b
t2=a-c t3=t1+t2 t4=t3+t2
d=t4

5. **What is meant by peephole optimization?**

Peephole optimization is a technique used in many compliers, in connection with the optimization of either intermediate or object code. It is really an attempt to overcome the difficulties encountered in syntax directed generation of code.

6. **What are the issues in the design of code generators?**

Input to the code generator Target programs
Memory management Instruction selection Register allocation
Choice of evaluation order Approaches to code generation

7. **What is register descriptor and address descriptor?**

A register descriptor keeps track of what is currently in each register.
An address descriptor keeps track of the location where the current value of the name can be found at run time.

8. **Define DAG.**

A DAG for a basic block is a directed acyclic graph with the following labels on nodes:
Leaves are labeled by unique identifiers, either variable names or constants.
Interior nodes are labeled by an operator symbol.
Nodes are also optionally given a sequence of identifiers for labels.

9. Name the techniques in Loop optimization.
Code Motion, Induction variable elimination, Reduction in strength

10. Define local optimization.
The optimization performed within a block of code is called a local optimization.

11. Define constant folding.
Deducing at compile time that the value of an expression is a constant and using the constant instead is known as constant folding.

12. What is code motion?
Code motion is an important modification that decreases the amount of code in a loop.

13. What are the properties of optimizing compilers?
Transformation must preserve the meaning of programs. Transformation must, on the average, speed up the programs by a measurable amount
A Transformation must be worth the effort.
The source code should be such that it should produce minimum amount of target code.
There should not be any unreachable code.
Dead code should be completely removed from source language.

14. Define Local transformation & Global Transformation.
A transformation of a program is called Local, if it can be performed by looking only at the statements in a basic block otherwise it is called global.

15. What is meant by Common Sub-expressions?
An occurrence of an expression E is called a common sub-expression, if E was previously computed, and the values of variables in E have not changed since the previous computation.

16. What is meant by Dead Code? Or Define Live variable?
A variable is live at a point in a program if its value can be used subsequently otherwise, it is dead at that point. The statement that computes values that never get used is known Dead code or useless code.

17. What is meant by Reduction in strength?
Reduction in strength is the one which replaces an expensive operation by a cheaper one such as a multiplication by an addition

18. What is meant by loop invariant computation?
An expression that yields the same result independent of the number of times the loop is executed is known as loop invariant computation.

19. When is a flow graph reducible?
A flow graph is reducible if and only if we can partition the edges into two disjoint groups often called the forward edges and back edges.

20. What is induction variable?

A variable is called an induction variable of a loop if every time the variable changes values, it is incremented or decremented by some constant.

UNIT V
PART B

1. Explain briefly about the principal sources of optimization
2. Explain in detail about optimization of basic blocks.
3. Construct the DAG for the following Basic block & explain it.

```
1. t1: = 4 * i
2. t2:= a [t1]
3. t3: = 4 * i
4. t4:= b [t3]
5. t5:=t2*t4
6. t6:=Prod+t5
7. Prod:=t6
8. t7:=i+1
9. i:= t7
10. if i<= 20 goto (1).
```

4. Discuss the following in detail

   (i)Semantic Preserving transformation
   (ii)Global Common subexpression

5. Write about the following in detail

   (i)copy propagation (ii)Dead code Elimination
   (iii)code motion

6. Analyze Peephole optimization with suitable examples
7. Discuss in detail about the Use of Algebraic Identities.
8. Describe in detail about the flow of control optimization.
9. Describe about induction variable and end reduction in strength
10. Describe the efficient data flow algorithms in detail.
11. Give an example to identify the dead code in the DAG.
12. Describe the representation of array using DAG with example.
13. Create DAG and three – address code for the following program.

```
 i = 1; s = 0;
while ( i<= 10)
   {
     s = s+ a[i] [i];
     i = i + 1;
}
```