GATE

2019

DATA STRUCTURE & ALGORITHM (INCLUDING C)

COMPUTER SCIENCE

ECG Publications

ECG Publications

A Unit of **ENGINEERS CAREER GROUP**

Head O ce: S.C.O-121-122-123, 2nd oor, Sector-34/A, Chandigarh-160022

Website: www.engineerscareergroup.in **Toll Free:** 1800-270-4242

E-Mail: ecgpublica ons@gmail.com | info@engineerscareergroup.in

GATE-2019: Data Structure & Algorithm (including C)| Detailed theory with GATE previous year papers and detailed solu ons.

First Edi on: 2016

Price of Book: *INR 420/-*

CONTENTS

SECTION-A (PROGRAMMING IN C)

SECTION-B (DATA STRUCTURE)

SECTION-C (ALGORITHM)

CHAPTER		**PAGE**

SECTION-A

(PROGRAMMING IN C)

CHAPTER - 1
BASICS

1.1 INTRODUCTION
C is a remarkable language. Originally designed by Dennis Ritchie, working at AT & T Bell laboratories in New Jersey.
C is a structured language. It allows verity of programs in small modules. It is easy for debugging, testing and maintenance if a language is a structured one.
C have a low level access to memory, simple set of keywords and clean style. Many later languages have borrowed syntax directly or indirectly from C language. For example Java, PHP, Java Script and many other languages are mainly based on C language. C^{++} is nearly a superset of C language.

1.2 STRUCTURE OF C PROGRAM
```
Include header file section
Global declaration section
Main ( )
{
    Declaration Part
    Executable Part
}
User defined functions
{
    Statements
}
```
Let's write our first C Program (ECG.C)
```
# include < stdio.h >
int main (void)
    {
        Print f (''Engineers Career Group'');
        Return o;
    }
```
Let’s Analyze the Program line by line

1. **Include < Stdio.h >**
All the lines starting with # are processed by preprocessor. Preprocessor is a program that is called by complier. In another words we can say preprocessor will take input program given by programmer and will produces output program where, there are no lines starting with #, all those lines processed by preprocessor.
Here, # include <stdio.h> will be replaced by file ‘stdio.h’ which declares the print f( ) function, by preprocessor.

2. Int Main (void)
In C program, there is a fixed point from where execution of complied c program begins, i.e. main ( ) function int written before main ( ) indicates return type of main ( ) and (void) indicates that main ( ) function doesn’t take any parameters.

GATE QUESTIONS

1. Consider the following C program:

```
#include<stdio.h>
int counter = 0;
int calc (in a, int b) {
     int c;
     counter ++;
     if (b =  = 3) return (a*a*a);
     else  {
        c=calc (a, b/3);
        return (c*c*c);
   }
}
int main ( )  {
    calc (4, 81);
    print ("%%d", counter);
}
```

The output of this program is _______.

(GATE - 2018)

Sol. 1. (4)

2. Consider the following C program:

```
#include<stdio.h>
void fun 1 (char *s1, char *s2) {
   char  *temp;
   tmp = S1;
   s1 = s2;
   s2 = temp;
}
void fun2 (char **s1, char**s2) {
   char *tmp;
   tmp = *s1;
   *s1 = *s2;
   *s2= tmp;
}
int main ( ) {
   char *str1 = "Hi", *str2 = "Bye",
   fun1 (str1, str2); print("%s %s", str1, str2);
   fun2 (&str1, &str2); print ("%s %s, str1,
str2);
   return 0;
}
```

The output of the program above is

(GATE - 2018)

(a) Hi Bye Bye Hi (b) Hi Bye Hi Bye
(b) Bye Hi Hi Bye (d) Bye Hi Bye Hi

Sol. 2.

CHAPTER - 2
FUNCTIONS, ARRAYS AND POINTERS

2.1 FUNCTION

In simple terms, we can say function is simply a series of statement that have been grouped together and given a name.

Functions are the building blocks of C programs and each function is essentially a small program with it's own declaration and statements. Every C program can be thought of as a collection of these functions.

Example.

```
# include <stdio.h>
Int average (int a, int b)
{
      return ((a+b)/2);
}
Int Main (void)
{
      int a =2, b = 4;
int C = average (a, b);
Print f ("% d", c); return o;
}
```

Here, average () is a integer function i.e. the value returned by average () is of integer type. Average () taking a and b as input parameters (arguments) and finally returning the average value to the main (), who supplied the arguments or who called the average ().

It is not always compulsory that a function should return something i.e. if a function have data type "void" it will not return anything.

Example.

```
# include <stdio.h>
Void print_value (int n)
{
      Print f ("value is % d", n);
}
Int Main (void)
{
Int i;
For (i=20; i > 0; --i)
{ Print_value (i);}
Return o;
```

"i" will have value 20 initially, as soon as condition check, control goes into body and calls the print_value function.

In general we can define function as

```
              Return-type function (parameter)
{
                    Declarations
```

GATE QUESTIONS

1. Consider the following C program:

```
#include<stdio.h>
int counter = 0;
int calc (in a, int b) {
    int c;
    counter ++;
    if (b =  = 3) return (a*a*a);
    else  {
       c=calc (a, b/3);
       return (c*c*c);
   }
}
int main ( )  {
    calc (4, 81);
    print (“%%d”, counter);
}
```

The output of this program is _______.

(GATE - 2018)

Sol. 1. (4)

2. Consider the following C program:

```
#include<stdio.h>
void fun 1 (char *s1, char *s2) {
   char  *temp;
   tmp = S1;
   s1 = s2;
   s2 = temp;
}
void fun2 (char **s1, char**s2) {
   char *tmp;
   tmp = *s1;
   *s1 = *s2;
   *s2= tmp;
}
int main ( ) {
   char *str1 = “Hi”, *str2 = “Bye”,
   fun1 (str1, str2); print(“%s %s”, str1, str2);
   fun2 (&str1, &str2); print (“%s %s, str1,
str2);
   return 0;
}
```

The output of the program above is

(GATE - 2018)

(a) Hi Bye Bye Hi
(b) Hi Bye Hi Bye
(b) Bye Hi Hi Bye
(d) Bye Hi Bye Hi

Sol. 2.

CHAPTER - 3
STRINGS, STRUCTURE AND UNION

3.1 STRINGS
Group of characters can be stored in a character array, some times also called as string.
Char a [ ] = {‘G’, ‘A’, ‘T’, ‘E’, ‘’\ o’}
Here, each character will take 1 byte of memory and ‘\ o’ is called as null character i.e. termination of sequence.
The above array declaration is similar to
Char b [ ] – “GATE”;
Here, GATE is a string literal, i.e. characters are enclosed between double quotes. In this ‘\o’ will be added in the end automatically by compiler.

3.1.1. In essence, C treats string literals as character arrays. When a C complier encounters a string literal of length n in a program, it sets aside n+1 bytes of memory for the string. i.e.

G	A	T	E	\0

3.1.2. When C allows char*, then we can use string literals. For example
Char * p = “GATE”
This assignment doesn’t copy the characters in “GATE”, rather makes P a pointer points to first character.

3.2. CHARACTER ARRAY VERSUS CHARACTER POINTERS
1. Char a [ ] = “GATE 2018”
2. Char * b = “GATE 2018”;
In (1), a is an array, while in (2) b is a pointer

Let’s chck some operation on both

a [o] = ‘a’; // valid	P [o] = ‘b’; // invalid
a [3] = ‘r’; // valid	P [2] = ‘r’; // invalid
a [12] = ‘z’ ;// invalid	P [5] = ‘q’ //invalid

3.3 STANDARD LIBRARY STRING FUNCTION

Function	Use
Strlen	Finds the length of string
Strcat	Appends one string at the end of another
Strncat	Appends first n characters of a string at the end of another
Strcpy	Copies a string into another
Strncpy	Copies first n characters of a string into another
Strcmp	Compares two strings
Strcmpi	Compare two strings regardless to case
Strrev	Reverse string
Strdup	Duplicate a string

GATE QUESTIONS

1. Consider the C program fragment below which is meant to divide x by y using repeated subtractions. The variables x, y. q and r are all unsigned int.

```
while (r > = y)
{
        r = r = – y;
        q = q + 1;
}
```

Which of the following conditions on the variable x, y, q and r before the execution of the fragment will ensure will ensure that the loop terminates in a state satisfying the condition x = = (y*q +r)?

[GATE - 2017]

(a) (q = = r) && (r = = 0)
(b) (x > 0) && (r = = x) && (y > 0)
(c) (q = = 0) && (r = = x) && (y > 0)
(d) (q = = 0) && (y > 0)

2. Consider the following C Program.

```
#include <stdio.h>
int main ()
{
    int m = 10;
    int n, nl;
    n = ++m;
    nl = m++;
    n – –;
    – – nl;
    n – = nl;
     printf("%d", n);
    return 0;
}
```

The output the program is _____.

[GATE - 2017]

3. Consider the following C Progarm.

```
#include <stdio.h>
#include <string.h>
int main()
{
     char *c = " GATECSIT2017";
     char *p = c;
     print("%d", (int) strlen (c+2 [p] –6 [p]– 1));
     return 0;
}
```

The output of the program is ____.

[GATE - 2017]

4. Match the following.

A. static char var;
B. m = malloc (10);
   m = NULL;
C. char *ptr [10];
D. register int var1;

(i)Sequence of memory locations to store addresses
(ii)A variable located in data section of memory
(iii)Request to allocate a CPU register to store data
(iv)A lost memory which cannot be freed

[GATE - 2017]

(a) A-ii, B-iv, C-i, D-iii
(b) A-ii, B-I, C-iv, D-iii
(c) A-ii, B-iv, C-iii, D-i
(d) A-iii, B-iv, C-I, D-ii

5. Consider the following function implemented in C.

```
void  printxy (int x, int y)
{
    int *ptr;
    x = 0;
    ptr = &x;
    y = *ptr;
    *ptr = 1;
    print ("%d, %d", x, y);
```

The output of invoking printxy (1, 1) is

[GATE - 2017]

(a) 0, 0 (b) 0, 1
(c) 1, 0 (d) 1, 1

6. Consider the C function foo and bar given below.

```
    int foo (int val)
```

ASSIGNMENT

1. We require a code to print the integer values from 0 to 9. Identify the error?

```
#include<stdio.h>
#include<conio.h>
Void main ( )
{
    static int I,
    clrscr ( ) ;
    for (;i<=9;) ;
    {
        Printf("/n value is: %d", i);
    }
    getch( );
}
```

2. Find the output of the following code?

```
#include<stdio.h>
#include<conio.h>
void main ( )
{
    int i;
    printf ("Enter the value of i = ");
    sanf ("%d", &i) ; /* input : 5 */
    clrscr ( );
    do
    {
        i++;
        Printf ("\nHello User");
    }
    while (i<4) ;
    getch ( ) ;
}
```

3. Find out the error in the following code.

```
#include<stdio.h>
#include<conio.h>
void main( )
{
    int i=1, j;
    clrscr ( ) ;
    do
    {
        for (j=1 ;; j++)
        {
            if (j>2)
                break ;
            if (i = j)
                continue;
            printf ("\n%d%d", i, j) ;
        }
        i++;
        while (1<3)
    }
    getch ( ) ;
}
```

4. What will be the value of I and j after execution?

```
for (i=0, j=0; i<5, j<25, i++, j++) ;
printf ("i=%d j=%d ", i, j);
```

5. What is the output of the following code?

```
#include<stdio.h>
void main( )
{
    int var = −10;
    for (;var;printf("%d\n", var++));
}
```

6. What is the output of the following code?

```
#include<stdio.h>
main ( )
{
    int var=0;
    for (;var++;printf ("%d", var)) ;
    printf ("%d", var) ;
}
```

7. What is the output of the following code?

```
#include<stdio.h>
main ( )
{
    int y;
    scanf ("%d", &y) ;
    /* input given is 2000 */
```

SECTION-B

(DATA STRUCTURE)

CHAPTER - 1
ARRAYS

1.1 INTRODUCTION
1. Arrays are collection of homogeneous data.
2. Array elements are stored in successive memory locations in the memory.
3. Arrays are broadly categorized as follows
(i) One-Dimensional Array
(ii) Multi-Dimensional Array

1.1.1 One-Dimensional Array
In 1-D array, the elements are stored at consecutive locations.

10	12	14	16	18	20	22	24	26
0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**

Lower Bound **(Indices)** **Upper Bound**

Lower Bound (LB): Smallest index of an element in the array.
Upper Bound (UB): Largest index of an element in the array.
Length of the Array = No. of elements in the Array
Length of the Array = UB – LB + 1

> Lower Bound of Array is taken as '0' until and unless not mentioned.

1.1.1.1 Address Calculation in One-Dimensional Array
Physical address of any element of array is computed as follows
General Formula
Physical address of a[i] = B.A+ (i – LB of array) Size of the element
Where, B.A (Base address) = Physical address of starting element in the array

1.1.2 Two-Dimensional Array
1. It is an instance of Multi-Dimensional Array.
2. It is collection of 1D arrays.
3. Notation used to represent 2D array is array name[number of 1D arrays][number of elements in each 1D array) **or**
Array name [index vector of rows][index vector of columns]
4. In Mathematics, matrices can be treated as 2D array, where numbers of rows are same as number of 1D arrays and number of columns are same as no: of elements in each 1D array.

1.1.2.1 Graphical Representation

$$\begin{vmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{vmatrix}$$

GATE QUESTIONS

1. Consider the following snippet of a C program. Assume that swap (&x, &y) exchanges the contents of x and y.

```
int main ( )
{
   int array [ ] = {3, 5, 1, 4, 6, 2};
   int done = 0;
   int i;
   while (done == 0)
   {
         done = 1;
         for (i=0; i<= 4; i++)
         {
           if (array [i] < array [i + 1])
           {
              swap (&array[i], &array [i +1]);
              done = 0;
           }

         }
         for (i=5; i>=1; i– –)
         {
           if (array[i] > array[i –1])
           {
           swap(&array[i], &array[i–1];
           done = 0;
           }
         }
   }
   printf("%d", array [3]);
 }
```

The output of the program is ________.

[GATE - 2017]

2. What is the output of the following C code? Assume that the address of x is 2000 (in decimal) and an integer requires four bytes of memory.

```
int main( )
{ unsigned int x [4][3] =
{{1, 2, 3}, {4, 5, 6}, {7, 8, 9}, {10, 11, 12}};
printf("%u,%u, %u", x+3, *(x+3),*(x+2)+3);
}
```

[GATE - 2015]

(a) 2036, 2036, 2036
(b) 2012, 4, 2204
(c) 2036, 10, 10
(d) 2012, 4, 6

3. Consider the following C program segment.

```
# include <stdio.h>
int main()
{
Char sl[7] = "1234", *p;
p = sl + 2;
*p = '0';
printf("%s", sl)
}
```

What will be printed by the program?

[GATE - 2015]

(a) 12 (b) 120400
(c) 1204 (d) 1034

4. Consider the following C program.

```
# include <stdio.h>
int main ()
{
static int a[ ] = {10, 20, 30, 40, 50};
static unit *p[ ] = {a, a + 3, a + 4, a + 1, a + 2};
int **ptr = p;
ptr++;
printf("%d%d", ptr-p, **ptr);
}
```

The output of the program is ______.

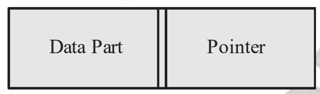
[GATE - 2015]

5. Suppose c =(c[0],……c[k−1]) is an array of length k, where all the entries are form the set {0, 1}. For an positive integers a and n, consider the following pseudo code.

DO so METHING (c, a, n)
$z \leftarrow 1$
for I $\leftarrow$ 0 to k − 1
do $z \leftarrow z^2$ mod n
if c[i] = 1
then $z \leftarrow (z \times a)$ mod n
return z

CHAPTER - 2
LINKED LIST

2.1 INTRODUCTION
1. It is collection of data elements, called nodes. Each node has its data part and pointer to maintain the order of nodes.
2. It is used to store and remove the data dynamically as per requirement.

2.1.1 Representation of Node


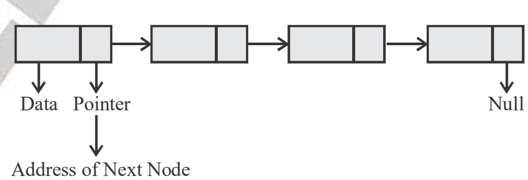
Node

2.1.2 Need of Linked List instead of Array
In array, there is wastage of memory whenever the number of stored elements are less than the maximum size of the array. But in linked list, only required number of elements are allocated space in the memory, extra unused space is not allocated. Maximum number of insertions can be done in linked list until memory overflows.

> Space utilization is efficient in Linked list.

2.2 TYPES OF LINKED LIST
1. Single Linked List
2. Circular Linked List
3. Double Linked List
4. Circular Double Linked List

2.2.1 Single Linked List
1. It contains nodes having single pointer that contains address of next node in the list.
2. It can be represented as follows



3. We use structure to define a node in linked list as follows:

```
struct node
{
int data;                // used to store the data elements of the linked list
struct node * link;      // used to store the pointer of next data element in the Linked list.
};
```

GATE QUESTIONS

1. Consider the C code fragment given below.

```
Typedef struct node
{ int data;
Node * next;
} node;
Void joint (node * m , node * n)
{
Node * p = n'
While (p → next ! = NULL)
{
p = p → next;
}
p → next = m;
}
```

Assuming that m and n point to valid NULL-terminated linked lists, invocation of joint will

[GATE - 2017]

(a)Append list m to the end of list n for all inputs.

(b)Either cause a null pointer dereference or append list m to the end of list n.

(c)Cause a null pointer dereference for all inputs.

(d)Append list n to the end of list m for all inputs.

2. N items are stored in a sorted doubly linked list. For a delete operation, a pointer is provided to the record to be deleted. For a decrease-key operation, a pointer is provided to the record on which the operation is to be performed. An algorithm performs the following operations on the list in this order: $\Theta(N)$ delete, $O(\log N)$ insert, $O(\log N)$ find, and $\Theta(N)$ decrease-key. What is the time complexity of all these operations put together?

[GATE - 2016]

(a) $O(\log^2 N)$ (b) $O(N)$

(c) $O(N^2)$ (d) $\Theta(N^2 \log N)$

3. The following C function takes a single-linked list as input argument. It modifies the list by moving the last element to the front of the list and returns the modified list. Some part of the code is left blank.

```
typedef struct node
{
    int value;
    struct node *next;
} node;
Node *move_to-front (Node *head)
{
    Node *p, *q;
    if ((head == NULL) | | (head – > next = = NULL))
       return head;
    q = NULL;
    p = head;
    while (p – >next ! = NULL)
    {
          q = p;
          p = q – > next;
    }
    ___________________
    return head;
}
```

Choose the correct alternative to replace the blank line.

[GATE - 2010]

(a) q = NULL; p → next = head; head = p;

(b) q→ next = NULL; head = p; p → next = head;

(c) head = p; p → next = q; q → next = NULL;

(d) q→ next = NULL; p → next = head; head = p;

4. The following C function takes a singly – linked list of integers as a parameter and rearranges the elements of the list. The function is called with the list containing the integers 1, 2, 3, 4, 5, 6, 7 in the given order. What will be

CHAPTER - 3
STACKS

3.1 INTRODUCTION

1. It is linear list in which data elements are inserted and deleted at one end.
2. It uses LIFO (Last In First Out) technique to access each element.

Example. Pile of books, pile of coins.
To take the book presents at the bottom of the pile, we firstly, remove all the books above that bottom book in order. And to add new book in the pile, it is added at the top not in the middle. That is called LIFO technique. Similar is the case with pile of coins.

3.2 ABSTRACT DATA TYPE (ADT) OF STACK

Stack Definition contains information about accessing technique used in stack, Standard operations and Status operations, Pointers. It has
Accessing Technique: LIFO(Last In First Out)
TOP: It is a pointer that points to recently pushed element.
Standard Operations : Pop ( ), Push ( )
Status Operations : Isempty ( ), Isfull ( )

3.2.1 PUSH() Operation

Let
S = Name of stack
TOP = Pointer pointing to top of the stack
N = size of stack // maximum number of elements that can be stored
x = element to be pushed

```
Void push(S, Top, N , x)
{
if (Top = = N – 1)
{
printf (``stack is full``);
exit (1) ;
}
else
Top ++;
S[Top] = x;
}
```

N-1 is the index for last element of stack, because stack starts from 0.

3.2.2 POP Operation

```
Pop (S, Top, N)
{
int y ;
```

GATE QUESTIONS

1. The attributes of three arithmetic operators in some programming language are given below.

Operator	Precedence	Associativity	Arity
+	High	Left	Binary
–	Medium	Right	Binary
*	Low	Left	Binary

The value of the expression $2-5+1-7*3$ in this language is _________.

[GATE - 2016]

2. Consider the following C function.

```
int fun (int n){
int x=1, k;
if (n==1) return x;
for  (k=1; k<n; ++k)
x = x + fun (k)  * fun (n–k);
return x;
}
```

The return value of fun (5) is ________

[GATE - 2015]

3. Consider the following recursive C function

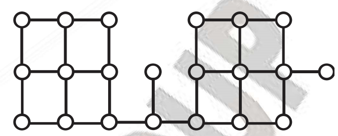
```
void get (int n)
{
If (n < 1) return
get (n – 1);
get (n – 3);
print f (“%d”, n);
}
```

If get (6) function is being called in main ( ) the how many times will the get () function be invoked before returning to the main ()?

[GATE - 2015]

(a) 15 (b) 25
(c) 35 (d) 45

4. Suppose depth first search is executed on the graph below starting at some unknown vertex that has not been visited earlier. Then the maximum possible recursion depth (including the initial call) is ________.



[GATE - 2014]

5. Suppose a stack implementation supports an instruction REVERSE, which reverses the order of elements on the stack, in addition to the PUSH and POP instructions. Which one of the following statements is TRUE with respect to this modified stack?

[GATE - 2014]

(a)A queue cannot be implemented using this stack.
(b)A queue can be implemented where ENQUEUE takes a single instruction and DEQUEUE takes a sequence of two instructions.
(c)A queue can be implemented where ENQUEUE takes a sequence of three instructions and DEQUEUE takes a single instruction.
(d)A queue can be implemented where both ENQUEUE and DEQUEUE take a single instruction each.

6. The following postfix expression with single digit operands is evaluated using a stack

8 2 3 ∧ / 2 3 * + 5 1 * –

Note that ^ is the exponentiation operator. The top two elements of the stack after the first* is evaluated are

[GATE - 2007]

(a) 6, 1 (b) 5, 7
(c) 3, 2 (d) 1, 5

7. An implementation of a queue Q, using two stacks S1 and S2, is given below

```
void insert (Q, x) {
    push(S1, x);
}
void delete (Q) {
    if (stack-empty(S2)) then
```

CHAPTER - 4
QUEUES

4.1 INTRODUCTION
It is linear list in which insertion of elements is done at the end and deletion is done at the front of the list.
It uses FIFO (First in First Out) technique to perform any operation.

Example. People waiting in line at a bank form a queue, where the first person in line is the first person to be waited on. New person cannot stand at any position in between queue, it will have to stand at the end of the queue.

4.1.1 Abstract Data Type (ADT) of Queue
Definition: FIFO
1. FRONT = Pointer pointing to the element to be deleted.
2. REAR = Pointer pointing to the recently pushed element

4.1.2 Status Operations
1. PUSH ( )
2. POP ( )
3. Is empty ( )
4. Is full ( )

4.1.3 Graphical Representation of Queue
1. Enqueue
Insert an element in queue
2. Dequeue
Delete an element from the queue

1. If there is not a single element in the queue, then both the pointers point outside the queue.
2. If there is a single element in the queue, then both the pointers point at that element.

> It is not possible that one pointer is present inside the queue and other pointing outside the queue.
> Either both the pointers are present inside the queue or both will be outside the queue.

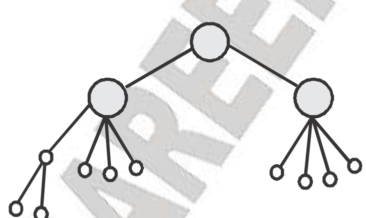
CHAPTER - 5
BINARY TREE

5.1 INTRODUCTION
In Binary tree, each node has at most two children.

5.1.1 Depth (Height) of Binary Tree
Maximum number of nodes in the longest branch.
Depth is one more than the largest level number of tree.

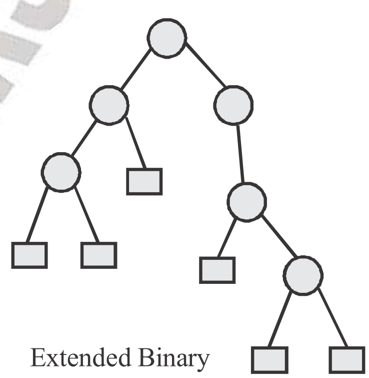
5.1.2 Types of Binary Tree
1. Complete Binary Tree
2. Extended Binary Tree

5.1.2.1 Complete Binary Tree
Binary Tree is said to be complete if all its levels, except possibly that last, have maximum number of possible nodes.
Depth(d_n) of the complete tree with n nodes is
$d_n = \log_2 n + 1$



Complete Binary Tree

5.1.2.2 Extended Binary Tree: 2-Tree
A Binary Tree is said to be a 2-Tree or an extended binary tree, if each node N has either 0 or 2 children. In such tree, nodes with 2 children are called internal nodes and nodes with 0 children are called external nodes.



Extended Binary

5.2 DEFINITION OF BINARY TREE NODE
1. data will store the information at the node.
2. lptr is the pointer to the left child of node.
3. rptr is the pointer to the right child of node.

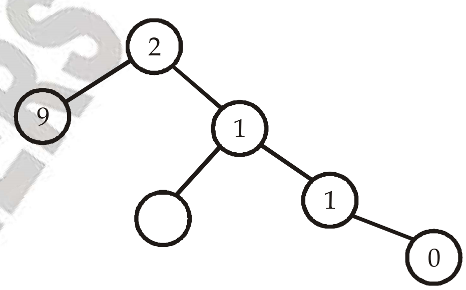
CHAPTER - 6
AVL TREES

6.1 INTRODUCTION
AVL trees are known as Balanced Binary Search Trees.

6.1.1 Why AVL?
Insertion of elements Z, X, Y,…………C, B, A in this order results in left skewed binary search tree.
Insertion of elements A, B, C,………X, Y, Z in this order results in right skewed binary search tree.
And disadvantage of a skewed binary search tree is that worst case complexity of a search is O(n). So, to reduce the searching time, we make binary tree of balanced height. And that balanced tree is called AVL tree.
dg

6.1.2 Balanced Factor
B.F. = Height of LST – Height of RST
or
B.F= Height of RST – Height of LST
Where, LST is left subtree and RST is right subtree.
A tree is considered as Balanced Tree, when balanced factor (B.F.) = 0, + 1 or – 1.
If balanced factor (B.F.) is anything else except 0, + 1 or – 1. Values, then it is considered as unbalanced tree.

Example. Unbalanced binary tree
Here, at the root node, the balanced factor comes out to be + 2, it is considered as an unbalanced tree.
(i) Balanced AVL



6.1.3 Searching in an AVL Search Tree
Searching an element in AVL tree is similar to binary search.

6.1.4 Insertion in an AVL Search Tree
Inserting an element into an AVL search tree in its first phase is similar to that of the one used in a binary search tree. However, if insertion of the element disturbs the balanced factor of any node in the tree. We use rotations technique, to restore the balance of the search tree.
To perform rotations it is necessary to identify a specific node A whose BF(A) is neither 0,1,-1, and which is the nearest ancestor to the inserted node on the path from the inserted node to the root.

GATE QUESTIONS

1. B+ Trees are considered **BALANCED** because

[GATE - 2016]

(a)The lengths of the paths from the root to all leaf nodes are all equal.
(b)The lengths of the paths from the root to all leaf nodes differ from each other by at most 1.
(c)The number of children of any two non-leaf sibling nodes differ by at most 1.
(d)The number of records in any two leaf nodes differ by at most 1.

2. What is the maximum height of any AVL-tree with 7 nodes? Assume that the height of a tree with a single node is 0.

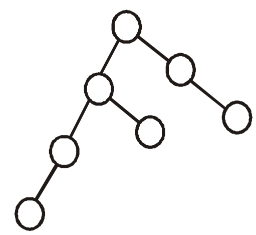
[GATE - 2009]

(a) 2 (b) 3
(c) 4 (d) 5

SOLUTIONS

Sol. 1. (a)
A Tree is balanced, if the length of the paths from the root to all leaf nodes are all equal.

Sol. 2. (b)
Maximum height of any AVL-tree with 7



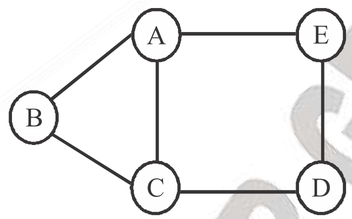
(There may be different way to draw AVL with 7 nodes)

CHAPTER - 7
GRAPHS

7.1 GRAPH
A graph has two properties

1. A set V of elements called nodes (vertices)

2. A set E of edges e in E is identified with a unique (unordered) pair [u, v] of nodes in V, denoted by e[u, v] , where u,v are the endpoints of edge e.



Graph (Figure 1)

7.1.1 Degree of Node
Number of edges containing that node.

In the above graph, vertex set V = {A, B, C, D, E}

Edge set E = {AB, AC, AE, BC, ED, CD, EC}

Degree of A (Degree (A)) = 3

Degree of B (Degree (B)) = 2

Degree of C (Degree (C)) = 3

Similarly we can find the degree of remaining nodes of the above graph.

7.1.2 Path
A sequence of nodes traversed to reach from one point to another.

7.1.2.1 Simple Path
A Path said to be simple if all the nodes are distinct with the exception that starting and ending vertex are distinct.

7.1.2.2 Closed Path
A Path said to be closed if all the nodes are distinct with the exception that starting and ending vertex are same.

7.1.3 Cycle
A cycle is a closed simple path with length 3 or more.

A cycle of length k is called k-cycle.

In the above figure 1,

Cycles: [A, B, C, E, A] and [A, C, D, E, A]

Path: (B, A, E) and (B, C, E) are simple paths of the length 2.

CHAPTER - 8
SORTING

8.1 RADIX SORT
1. It is the technique to sort elements.
2. It uses queue data structure to implement.

8.1.1 Steps for Radix Sort
1. Select the number in the list to be sorted with maximum digits (d).
2. Make all of the numbers having maximum number of digits by adding zeroes before the numbers, if required.
3. Arrange the numbers in the lists in the required order (increasing or decreasing) according
4. To the unit decimal place to $10^{(d-1)th}$ decimal place in each pass.

Example. 101, 1,79, 97, 86, 7, 44, 99, 421, 23, 49, 12
Solution.
Maximum no. of digits = 3
∴ Make all of them 3-digit
101, 001, 079, 097, 086, 007, 044, 099, 421, 023, 049, 012
1st Pass (Increasing order of digit at unit place)
101, 001, 421, 012,023, 044, 086, 097, 007, 079, 099, 049
2nd Pass (Ten's place digit)
101, 001, 007, 012, 421, 023, 044,049, 079, 086, 097, 099
3rd Pass (Hundred's Place digit)
001, 007, 012, 023, 044, 049, 079, 086, 097, 099, 101, 421
No. of passes = 3
Time complexity = O(m, n)
m = No of passes = Maximum number of digits of a number among the list's numbers.
n= no. of elements in the list to be sorted.
≈ O(n)

Example. 1, 2, 86,421,361,111,6,3,94,55,814,612,522,511
Solution.
Maximum number of digits = 3
∴ Make all of them = 3-digit

001,002,086,421,361,111,006,003,094,055,814,612,522,511
Pass-I
001,421,361,111,511,002,612,522,003,094,814,055,086,006
Pass-II
001,002,003,006,111,511,612,814,421,522,055,361,086,094
Pass-III
001,002,003,006,055,086,094,111,361,421,511,522,612,814
No. of passes = 3

8.2 BUCKET SORT
It is also a technique to sort the elements.

CHAPTER - 9
HASHING

9.1 DIFFERENT SEARCHING TECHNIQUES AND THEIR EFFICIENCY
1. The sequential (Linear) search algorithm takes time proportional to the data size, i.e, **O(n)**.
2. Binary search improves on linear search reducing the search time to **O(logn)**.
3. With a BST, an **O(logn)** search efficiency can be obtained; but the worst-case complexity is **O(n)**.
4.To guarantee the **O(log n)** search time, BST height balancing is required (i.e., AVL trees)

9.2 HASHING
1. It is the technique to access the records in the file fastly.
2. It has goal of Time complexity O(1).
3. It uses different hashing functions, which are applied on some field of the records. That field is called **hash field**.

9.2.1 Types of Hashing
1. Static Hashing
2. Dynamic Hashing

9.2.1.1 Static Hashing
1. In static hashing, the hash function maps search-key values to a fixed set of locations.
2. It uses single hash function
3. Its searching time is O(1)
4. It does not support range queries

(i) Problems of Static Hashing
1. There is fixed size of hash table due to fixed hash function
2. It may require rehashing of all keys when chains or overflow buckets are full

9.21..2 Dynamic Hashing
In dynamic hashing a hash table can grow to handle more items. The associated hash function must change as the table grows.
1. The load factor of a hash table is the ratio of the number of keys in the table to the size of the hash table.
2. The higher the load factor, the slower the retrieval.
3. With open addressing, the load factor cannot exceed 1. With chaining, the load factor often exceeds 1.

9.3 APPLICATIONS OF HASH TABLES
1. Database Systems
Specifically, it is for those that require efficient random access. Generally, database systems try to optimize between two types of access methods: sequential and random. Hash tables are an important part of efficient random access because they provide a way to locate data in a constant amount of time.

SECTION-C
(ALGORITHM)

CHAPTER - 1
ANALYSIS OF ALGORITHMS

1.1 INTRODUCTION
1. Algorithm is well defined sequence of computational steps that transform the input to output.
2. It can be designed to solve problems such as identifying all genes in human DNA, finding good routes to carry data from one machine to another machine to another machine, sorting, searching on particular data and many more.

1.2 EFFICIENCY
1. There are two valuable resources Memory and processor which are essentially required to solve any problem. Now a day, Processor′s speed is very good but they are not infinitely very fast and memory is also available in good sizes but it is not very cheap. So, processor's speed and memory sizes are two constraints for Algorithm efficiency. Another fact that can solve a given problem efficiently that is designing an algorithm efficiently.
2. Efficiency of algorithm can be defined in terms of time and space. It implies time and space taken by algorithm.
3. Time and space of given algorithm can vary on different input.
4. There are two broad categories of Analysis of algorithm
(i) Priory Analysis
(ii) Postinary Analysis.

(i) Priory Analysis
(a) It is machine independent that does not include processor′s speed and memory configuration of any system in efficiency of Algorithm.
(b) It defines the running time of an algorithm on a particular input is the number of primitive operations.

5. Each line of pseudocode takes constant time. One line of code can take different time (execution) than another line of code.

6. Any algorithm can have different running time on different inputs. So, each algorithm has lower and upper bounds on their running time on particular inputs.

7. Algorithm′s best performance is characterized by lower running time. It is called best case.

8. In worst case, Algorithm gives its worst performance that is upper bound on its running time.

9. Running time at all inputs except that of best and worst case, represents average case.

10. One algorithm is said to be more efficient then another if its worst case running time has a lower order of growth.

11. Each asymptotic running time is specified defined using different asymptotic notations.

12. Asymptotic running time means the running time of an algorithm on large input sizes.

ASSIGNMENT-I

Solve using substitution method

1. $T(n) = \begin{cases} 2T\left(\frac{n}{2}\right) + n & n > 1 \\ b & n \le 1 \end{cases}$

2. $T(n) = \begin{cases} 2T\left(\frac{n}{2}\right) + 2 & n > 2 \\ 1 & n \le 2 \end{cases}$

3. $T(n) = \begin{cases} 8T\left(\frac{n}{2}\right) + n^2 & n > 2 \\ 1 & n \le 2 \end{cases}$

4. $T(n) = \begin{cases} 2T(n-1) + c & n > 1 \\ 1 & n \le 1 \end{cases}$

5. $T(n) = \begin{cases} T(n-1) + n & n > 1 \\ 1 & n \le 1 \end{cases}$

Solve using master method

6. $T(n) = 2T\left(\frac{n}{2}\right) + 2$

7. $T(n) = 5T\left(\frac{n}{2}\right) + n$

8. $T(n) = 7T\left(\frac{n}{2}\right) + n^2$

9. $T(n) = 27\,T\left(\frac{n}{3}\right) + n^3$

CHAPTER - 2
DIVIDE AND CONQUER

2.1 DIVIDE AND CONQUER
1. It is another way to design algorithms.
2. It firstly divides the problems into number of independent sub problems.
3. Then, it solves the sub problems by solving them recursively
4. It combines the sub problems solutions to give a solution to the original problem.

2.1.1 Control Abstraction of Divide and Conquer
Abstraction is used because some functions are not defined such as small, divide, solution etc.
If P is the problem, DAC works as follows

```
DAC (P)
{
if (small (P))   //checks whether problem is small or big
return (S(P))
else
K = divide (P) into k parts
return (combine (DAC(P1), DAC (P2), DAC (P3)……..DAC (Pk)))
}
```

2.2 APPLICATIONS OF DIVIDE AND CONQUER
1. Finding maximum and minimum
2. Binary search
3. Quick Sort
4. Merge Sort
5. Selection Procedure

2.2.1 Finding Maximum and Minimum
It is the problem to find maximum and minimum number among n numbers.
There are two approaches to solve this problem
1. Straight Method
2. Divide and Conquer

1. Straight Method
(i) This method finds maximum and minimum by comparing all remaining elements with first element of the array linearly.
(ii) Its algorithm is following

```
straight max-min (a, n)
{
max ← a[1]
min ← a[1]
for (i = 2 to n)
if max < a[i] then
do max ← a[i]
else
```

CHAPTER - 3

GREEDY TECHNIQUE, DYNAMIC PROGRAMMING

3.1 INTRODUCTION
1. Greedy and Dynamic Techniques are other approaches to design algorithms.
2. Some algorithms are efficient if they are designed using these approaches.
3. Every problem has its Definition, Solution space, Feasible Solution space, Optimal Solution space.

3.1.1 Problem Definition
Knowing the problem, its inputs and outputs and its conditions clearly.
1. Solution Space: All possible solutions of the problem over given input conditions
2. Feasible Solution Space: It includes solutions from the solution space which satisfies the conditions of the problem.
3. Optimal Solution Space: It includes solutions from the feasible solutions, which satisfy the optimal conditions of the problem.

3.2 GREEDY TECHNIQUE
1. Algorithms that use greedy technique are called greedy algorithm.
2. This technique allows to algorithm to make the choice that always looks the best choice at the moment. It implies it makes locally optimal choice in the hope that this choice will lead to a globally optimal solution.
3. It is used in various optimization problems.
4. But it does not yield optimal solutions for some problems.

3.2.1 Control Abstraction of Greedy Technique
a is an array of n object. Initially solution is ϕ

```
Greedy (a, n)
{
Solution = ϕ
For (i = 1 to n)
{
xi = select (n);
it (feasible (xi)
Add (solution, xi);
}
}
```

3.3 APPLICATION OF GREEDY TECHNIQUE
1. Job sequencing with deadline
2. Real knapsack (fractional knapsack)
3. Optimal merge pattern
4. Minimum cost spanning tree
5. Huffman coding
6. Single Source Shortest Path

CHAPTER - 4
COMPLEXITY CLASSES

4.1 INTRODUCTION
1. There are many problems exist in the world. Some of the problems are very easy and some are difficult. Easy problems are also called solvable and difficult problems are those problems which are not solvable or take more time to solve.
2. Solvable problems are called tractable problems.

4.2 ABSTRACT PROBLEM
1. It is defined as binary relation on a set I of problem instances and a set S of problem solutions.
2. Abstract decision problem is a function that maps the instance set I to the solution set {0, 1}.
For example, decision problem is related to shortest-path is the Problem path.
i = < G, u, v, k > is the instance of the shortest path problem that belongs to set I of shortest path.
If path (i) = yes, it implies there is a path from u to v has almost k edges. Otherwise path (i) = No.

4.3 ENCODING PART
1. It is a mapping of abstract objects from a set to the set of binary strings such as set N = {0, 1, 2, 3, 4, …} $\Rightarrow$ e (3) = 11.
2. Similarly are abstract objects such as polygons, graphs, functions, ordered pairs, programs can be encoded as binary strings.
3. Encoding also exists in shortest part abstract decision problem where every instance from set S can be encoded
4. It transforms abstract problem to concrete problem.
5. The computer algorithm that solves abstract decision problem actually takes on encoding of a problem instance as input.
6. Concrete problem has input instances as binary strings.
7. Polynomial-time solvability of a problem also depends upon encoding but it is assumed that it is independent of encoding procedure.
8. Theory of computation discipline allows us to express the relation between decision problems and algorithms that solve them concisely.
9. If there is an abstract decision problem with instance set I, its encoding set e(I) and solution set S = {0, 1}. Then, if an algorithm/machine model accepts a string $x \in e(I)$ if I given as input then language (L) of machine/Algorithm will be L ={ $x \in e(I)$: S(x) = 1 }. So, it includes all accepted strings but it rejects $x \in e(I)$ and S(x) = 0
10. Language L/problems is said to be decidable if every binary string in L is accepted by machine/algorithm and every binary string into in L is rejected by the machine/algorithm. Therefore, all Turing machine problems/languages are decidable.
11. A language L is said to be decided in polynomial time, if there is an algorithm for which a constant k exist and for strings of any-length n $x \in \{0, 1\}^*$, the algorithm correctly decides whether $x \in L$ in time O (n^k).
12. Turing machine languages are decided in finite amount of time. It also implies that they are decidable
13. Some algorithm/machine accepts all $x \in L$, but loop forever. If $x \notin L$. These languages are called recursive enumerable.

GATE QUESTIONS

1. Consider two decision problems Q_1, Q_2 such that Q_1 reduces in polynomial time to 3-SAT and 3-SAT reduces in polynomial time to Q_2. Then which one of the following is consistent with the above statement?

[GATE - 2015]

(a) Q_1 is in NP, Q_2 is NP hard.
(b) Q_2 is in NP, Q_1 is NP hard.
(c) Both Q_1 and Q_2 are in NP.
(d) Both Q_1 and Q_2 are NP hard.

2. Consider the following statements.
I. The complement of every Turing decidable language is Turing decidable
II. There exists some language which is in NP but is not Turing decidable
III. If L is a language in NP, L is Turing decidable
Which of the above statements is/are true?

[GATE - 2015]

(a) Only II (b) Only III
(c) Only I and II (d) Only I and III

3. Language L1 is polynomial time reducible to language L2. Language L3 is polynomial time reducible to L2, which in turn is polynomial time reducible to language L4.
Which of the following is/are true?
I. If $L4 \in P$, $L2 \in P$
II. If $L1 \in P$ or $L3 \in P$, then $L2 \in P$
III. $L1 \in P$, if and only if $L3 \in P$
IV. If $L4 \in P$, then $L1 \in P$ and $L3 \in P$

[GATE - 2015]

(a) II only (b) III only
(c) I and IV only (d) I only

4. Let π_A be a problem that belongs to the class NP. Then which one of the following is TRUE?

[GATE - 2009]

(a) There is no polynomial time algorithm for π_A
(b) If π_A can be solved deterministically in polynomial time, then P = NP
(c) If π_A is NP – hard, then it is NP-complete
(d) π_A may be undecidable

5. The subset sum problem is defined as follows: Given a set S of n positive integers and a positive integer W; determine whether there is a subset of S whose elements sum to W.
An algorithm Q solves this problem in O(nW) time. Which of the following statements is false?

[GATE - 2007]

(a) Q solves the subset sum problem in polynomial time when the input is encoded in unary
(b) Q solves the subset sum problem in polynomial time when the input is encoded in binary
(c) The subset sum problem belongs to the class NP
(d) The subset sum problem is NP-hard

6. Let S be an NP-complete problem Q and R be two other problems which are known to be in NP. Q is polynomial-time reducible to S and S is polynomial-time reducible to R. Which one of the following statements is true?

[GATE - 2006]

(a) R is NP-complete (b) R is NP-hard
(c) Q is NP-complete (d) Q is NP-hard

GATE

2019

THEORY OF COMPUTATION & COMPILER DESIGN

COMPUTER SCIENCE

ECG Publications

A Unit of **ENGINEERS CAREER GROUP**

Head O ce: S.C.O-121-122-123, 2nd Floor, Sector-34/A, Chandigarh-160022

Website: www.engineerscareergroup.in **Toll Free:** 1800-270-4242

E-Mail: ecgpublica ons@gmail.com | info@engineerscareergroup.in

GATE-2019: Theory of Computa on & Compiler Design| Detailed theory with GATE previous year papers and detailed solu ons.

First Edi on: 2016

Price of Book: *INR 300/-*

CONTENTS

SECTION-A (THEORY OF COMPUTATION)

SECTION-B (COMPILER DESIGN)

SECTION - A
THEORY OF COMPUTATION

CHAPTER - 1
FINITE AUTOMATA

1.1 INTRODUCTION

1. Theory of computation is a model of digital computer which does not consider platform dependent aspects of the computer.

2. It is a model or Pseudo code to understand computation.

3. Each automaton has the following characteristics:

(i) Input
(ii) Output
(iii) States
(iv) States relation
(v) Output relation

(i) Input: It is the set of possible inputs that can be applied on input side of model of automaton.

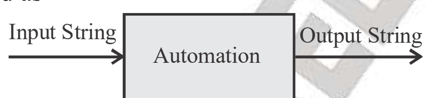
(ii) Output: It is the set of possible outputs of automaton.

(iii) States: It is set of possible states in which an automaton can be at any instant.

(iv) State Relation: It defines how different states are achieved, which is determined by present inputs and present states.

(v) Output Relation: The output is related to either state only or both the input and the state.

An automaton can be modeled as



1.2 BASIC DEFINITIONS

1. Alphabet

Any finite non- empty set of symbols, denoted by sigma(Σ)

Example. $\Sigma = \{a,b\}$

2. String

Any finite sequence of symbols over the given alphabet, denoted by (w)

Example. Let $\Sigma = \{0, 1\}$

Then strings are 0,1, 01, 10, 11,00, …….

3. Length of string (|w|)

It is defined as number of symbols in the string

Example.

(i) w =0 ,|w| = 1 (ii) w=1 , |w| =1 (iii) w =λ , |w| =0

4. Prefix of String

It is defined as sequence of leading symbols over the given string.

Example. Let w =TOC then prefix of strings are T, TO, TOC, λ.

5. Suffix of String

It is defined as sequence of trailing symbols over the given strings.

Example. Let w = TOC then suffixes of a string are C, OC, TOC , λ.

ASSIGNMENT

1. Consider the regular expression (0+1)(0+1)…..n times. The minimum state finite automation that recognizes the language by this regular expression contains:
(a) n states
(b) n + 1 states
(c) n + 2 states
(d) None of the above

2. Let $\Sigma = \{0, 1\}$, $L = \Sigma^*$ and $R = \{0^{n^2}\}$ such that n > 0} then language $L \cup R$ and R are respectively
(a) Regular, regular
(b) Non-regular, regular
(c) Regular, non- regular
(d) Non- regular, non- regular

3. The string 1101 does not belong to the set represented by
(a) 110* (0 + 1)
(b) 1 (0 + 1)* 101
(c) (10)* (01)* (00 + 11)*
(d) (00 + (11)* 01)*

4. Let L be the set of all binary strings whose last two symbols are the same. The number of states in the minimum state deterministic finite-state automation accepting L is
(a) 2 (b) 5
(c) 8 (d) 3

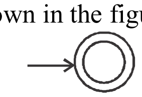
5. Which of the following is false?
(a) The languages accepted by FAs are regular languages
(b) Every DFA is an NFA
(c) There are some NFAs for which no DFA can be constructed
(d) If L is accepted by an NFA with ∈ transition then L is accepted by an NFA without ∈ transition.

6. How many minimum number of states are required in the DFA (over the alphabet {a, b}) accepting all the strings with the number of a's divisible by 4 and number of b's divisible by 5?
(a) 20 (b) 9
(c) 7 (d) 15

7. How many states does the DFA constructed for the set of all strings ending with "00" have?
(a) 2 (b) 3
(c) 4 (d) 5

8. How many minimum number of states will be there in the DFA accepting all strings (over the alphabet {a, b}) that do not contain two consecutive a's
(a) 2 (b) 3
(c) 4 (d) 5

9. The FSM shown in the figure accepts



(a) All strings (b) No strings
(c) ∈-alone (d) None of these

10. Consider the following transition table of FA

	δ	a	b
	state	q_1	q_0
→	q_0	q_1	q_0
	q_1	q_2	q_1
	q_2	q_3	q_2
	q_3	q_4	q_3
	(q_4)	q_4	q_4

What is true for the given FA?
(a) Accepts strings containing even number of a's and b's
(b) Does not accept strings containing b's
(c) Accepts strings independent of number b's
(d) Both (a) and (b)

GATE QUESTIONS

1. Consider the language L given by the regular expression (a+b)*b(a+b) over the alphabet {a, b}.The smallest number of states needed in a deterministic finite–state automation (DFA) accepting L is _____.

[GATE - 2017]

2. Let δ denote the transition function and $\hat{\delta}$ denote the extended transition function of the ε - NFA whose transition table is given below

δ	ε	a	b
$\rightarrow q_0$	$\{q_2\}$	$\{q_1\}$	$\{q_0\}$
q_1	$\{q_2\}$	$\{q_2\}$	$\{q_3\}$
q_2	$\{q_0\}$	ϕ	ϕ
q_3	ϕ	ϕ	$\{q_2\}$

Then $\hat{\delta}$ (a_2, aba) is

[GATE - 2017]

(a) ϕ
(b) $\{q_0, q_1, q_3\}$
(c) $\{q_0, q_1, q_2\}$
(d) $\{q_0, q_2, q_3\}$

3. The minimum possible number of states of a deterministic finite automation that accepts the regular language $L = \{w_1 a w_2 | w_1, w_2 \in \{a, b\}^*, |w_1| = 2\ |w_2| \geq 3\}$ is ______

[GATE - 2017]

4. Consider the following two statements:
I. If all states of an NFA are accepting states then the language accepted by the NFA is Σ*.
II. There exist a regular language A such that for all languages B, A∩B is regular. Which one of the following is **CORRECT**?

[GATE - 2016]

(a) Only I is true
(b) Only II is true
(c) Both I and II are true
(d) Both I and II are false

5. The number of states in the minimum sized DFA that accepts the language defined by the regular expression $(0+1)^*(0+1)(0+1)^*$ is ______.

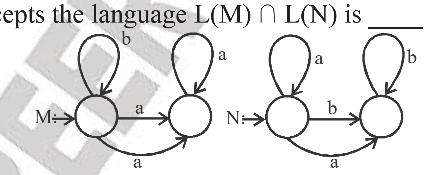
[GATE - 2016]

6. Which one of the following regular expressions represents the language: the set of all binary strings having two consecutive 0s and two consecutive 1s?

[GATE - 2016]

(a) $(0+1)^*\ 0011(0+1)^* + (0+1)^*\ 1100(0+1)^*$
(b) $(0+1)^*\ (00(0+1)^*\ 11+11(0+1)^*\ 00)(0+1)^*$
(c) $(0+1)^*\ 00(0+1)^* + (0+1)^*\ 11(0+1)^*$
(d) $00(0+1)^*\ 11+11(0+1)^*\ 00$

7. Consider the DFAs M and N given above. The number of states in a minimal DFA that accepts the language L(M) ∩ L(N) is ____



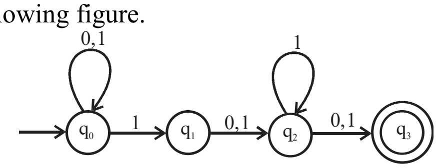
[GATE - 2015]

8. Let T be the language represented by the regular expression Σ*0011Σ* where Σ = {0, 1}. What is the minimum number of states in a DFA that recognizes L (complement of L)?

[GATE - 2015]

(a) 4
(b) 5
(c) 6
(d) 8

9. Consider the finite automation in the following figure.



What is the set of reachable states for the input string 0011?

[GATE - 2014]

(a) $\{q_0, q_1, q_2\}$
(b) $\{q_0, q_1\}$
(c) $\{q_0, q_1, q_2, q_3\}$
(d) $\{q_3\}$

10. Which of the regular expression given below represent the following DFA?

CHAPTER - 2
GRAMMARS, CONTEXT- FREE LANGUAGES

2.1 INTRODUCTION
1. Every language (such as English, French) has its corresponding grammar.
2. Grammar contains/describes the set of rules for the language.
3. It is useful for making translation easier using computer from one language to another.
4. A grammar can be described in mathematical way.
5. Firstly in 1956, Noam Chomsky gave a mathematical model of a grammar which is useful for computer languages.
6. Context–free grammar definition was being used in Backus–Naur form to describe ALGOL language.
7. Context-free languages are generated from context-free grammars (type-2).
8. They are applied in parser design.
9. They are also useful for describing block structure in programming languages.
10. These languages are accepted by Pushdown down Automata.

2.2 GRAMMAR
1. It defines the set of rules for a language.
2. It is defined by five tuples (V_N, Σ, P, S)
V_N is a finite nonempty set whose elements are called variables. Anything which can be substituted further (in upper case) is called variables/non-terminal
Σ is a finite non empty set whose elements are called terminals. Anything which cannot be substituted is called terminal/symbol.
$V_N \cap \Sigma = \phi$
S is a special variable from V_N called the start symbol
P is finite set having elements of form $\alpha \to \beta$ where α, β are string belong to $(V_N \cup \Sigma)^* \cdot \alpha$ should have at least one symbol from V_N. Its elements are called productions /production rules/rewriting rules.

Example.
G = (V_N, Σ, P, S) is a grammar where
V_N = {<sentence>, <noun>, <verb> <adverb>}
Σ = {Ram, somi, food, eat, dances, well}
S = <sentence>
and P contains following productions
< sentence > → <noun> <verb>
<sentence> → <noun> <verb> <adverb>
<noun> → Ram
<noun> → Somi
<verb> → eat
<verb> → dances
<adverb> → well

ASSIGNMENT

1. If G = {{S}, {a}, S, {S → SS}} find the language generated by G.
(a) L(G) = a*
(b) L(G) = a^+
(c) L(G) = ϕ
(d) Both (a) and (b)

2. Which of the following languages are context free?
$L_1 = \{a^m b^m c^n \mid m \geq 1 \text{ and } n \geq 1\}$
$L_2 = \{a^m b^m c^n \mid n \geq m\}$
$L_3 = \{a^m b^m c^m \mid m \geq 1\}$
(a) Only L_1
(b) L_1 and L_2
(c) Only L_2
(d) L_3

3. Which of the following definitions below generate the same languages as $L=\{x^n y^n | n \geq 1\}$
(i) E → xEy|xy
(ii) xy|(x^+ xyy^+)
(iii) $x^+ y^+$
(a) (i) only
(b) (i) and (ii)
(c) (ii) and (iii)
(d) (ii) only

4. If G is a context –Free grammar and w is a string of length n in L(G), how long is derivation of w in G, if G is in Chomsky normal form?
(a) 2n
(b) 2n + 1
(c) 2n – 1
(d) n

5. Which of the following is true?
(a) If language is context Free it can always be accepted by deterministic push-down automata
(b)The union of two context Free language is context Free
(c)The intersection of two context–Free language is context Free
(d)The complement of context-Free language is context Free language.

6. The grammar S → aaSbb|ab can generate the set
(a) $\{a^{2n+1} b^{2n+1}, n = 1, 2, 3,\ldots\}$
(b) $\{a^n b^n / n = 1, 2, 3, \ldots\}$
(c) $\{a^{2n+1} b^{2n+1} / n = 0, 1, 2, 3,\ldots\}$
(d) $\{a^{2n-1} b^{2n-1} / n = 0, 1, 2, 3,\ldots\}$

7. The language $a^m b^n c^{m+n}$ |m, n ≥ 1} is
(a) Regular
(b) Context free but not regular
(c) Context sensitive but not context free
(d) Type-0 but not context sensitive.

8. Consider the grammar G:
S → AB
A → aAA|∈
B → bBB|∈
If G_1 is constructed from G after eliminating the null productions, then G_1 is given by
(a) S → AB, A → aAA|aA|a, B → bBB|b
(b) S →AB|A|B|∈, A→ aAA|aA|a, B → bBB|bB|b
(c) S → AB|A|B, A → aAA|aA, B → bBB|bB
(d) S → AB, A → aAA|aA, B →bBB|bB

9. A grammar that is both left and right recursive for a non-terminal, is
(a) Ambiguous
(b) Unambiguous
(c) Information is not sufficient to decide
(d) None of these

10. Any string of terminals that can be generated by the following CFG satisfies which of the given choices?
S→XY, X→aX|bX|a, Y→Ya|Yb|a
(a) Has no consecutive a's or b's
(b) Has atleast two a's
(c) Has atleast one b
(d) None of these

11. Consider the language
$L_1 = \{a^n b^m c^n d^m \mid n \geq 1, m \geq 1$ and
$L_2 = \{a^n b^m c^m d^n \mid m \geq 1\}$
(a) Both L_1 and L_2 are context free
(b) L_1 is not context free but L_2 is context free

GATE QUESTIONS

1. Consider the following languages over the alphabet $\Sigma = \{a, b, c\}$
Let $L_1 = \{a^n b^n c^m | m, n \geq 0\}$ and $L_2 = \{a^m b^n c^n | m, n \geq 0\}$
Which of the following are context – free languages ?
I. $L_1 \cup L_2$
II. $L_1 \cap L_2$
[GATE - 2017]
(a) I only (b) II only
(c) I and II (d) neither I nor II

2. Consider the context –free grammar over the alphabet {a, b,c} given below .S and T are non-terminals
$G_1 : S \rightarrow aSb|T, T \rightarrow cT|\varepsilon$
$G_2 : S \rightarrow bSa|T, T \rightarrow cT|\varepsilon$
The language $L(G_1) \cap L(G_2)$ is
[GATE - 2017]
(a) Finite
(b) Not finite but regular
(c) Context – free but not regular
(d) Recursive but not context free

3. Consider the following grammar :
Stmt → if expr then expr else expr; stmt |o
Expr → term relop term |term
term → id | number
id →a|b|c
number → [0-9]
where relop is a relational operator (e.g., <, >,…), o refers to the empty statement, and if then , else are terminals.
Consider a program P following the above grammar containing ten if terminals .The number of control paths in P is _____.
For example, the program If e_1 then e_2 else e_3 has 2 control flow baths, $e_1 \rightarrow e_2$ and $e_1 \rightarrow e_3$
[GATE - 2017]

4. If G is a grammar with productions
$S \rightarrow SaS |aSb |bSa|SS|\varepsilon$
Where S is the start variable , then which one of the following strings is not generated by G ?
[GATE - 2017]
(a) abab (b) aaab
(c) abbaa (d) babba

5. Consider the following context-free grammar over the alphabet $\Sigma = \{a, b, c\}$ with S as the start symbol:
$S \rightarrow abScT|abcT$
$T \rightarrow bT |b$
Which one of the following represents the language generated by the above grammar?
[GATE - 2017]
(a) $\{(ab)^n(cb)^n - | n \geq 1\}$
(b) $\{(ab)\, cb^{m_1} cb^{m_2} ..cb^{m_n} | n, m_1, m_2, ... m_n \geq 1\}$
(c) $\{(ab)^n(cb^m)^n | m, n \geq 1\}$
(d) $\{(ab)^n(cb^n)^m | m, n \geq 1\}$

6. Identify the language generated by the following grammar , where S is the start variable
$S \rightarrow XY$
$X \rightarrow aX |a$
$Y \rightarrow aYb|\varepsilon$
[GATE - 2017]
(a) $\{a^m b^n | m \geq n, n > 0\}$
(b) $\{a^m b^n | m \geq n, n \geq 0\}$
(c) $\{a^m b^n | m > n, n \geq 0\}$
(d) $\{a^m b^n | m > n, n > 0\}$

7. Let L_1, L_2 be any two context –free languages and R be any regular language .Then which of the following is /are CORRECT?
[GATE - 2017]
I. $L_1 \cup L_2$ is context – free
II. $\overline{L_1}$ is context –free
III. $L_1 - R$ is context -free
IV. $L_1 \cap L_2$ is context- free
(a) I, II and IV only (b) I and III only
(c) II and IV only (d) I only

CHAPTER - 3
TURING MACHINE

3.1 INTRODUCTION

1. Turing machine is an automaton that fulfills two objective: Reorganization and computation.

2. It is generalization of pushdown automata that has tape of infinite length with head able to move in both directions or remain in the same position.

3.2 TURING MACHINE(TM)

It is an automation with the following properties:

1. Tape

It initially contains an input string. It can be potentially infinite on both sides, but the number of symbols written at any time on the tape is always finite.

2. Read - write Head

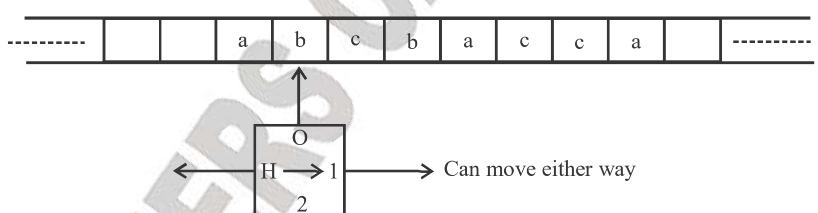
After reading the symbols on the tape and overwriting it with another symbol (which can be the same), the head moves to the next character, either on the left or on the right.

3. Finite Controller

It specifies the behavior of the machine for each state of the automaton and each symbol read from the tape, what symbol to write on the tape and which direction to move next.

4. Halting State

In addition to moving left or right, the machine may also halt. In this case, the turing machine is usually said to accept the input. Turing machine has only one halting (accepting state H.)



Model of Turing Machine

In this model , 0, 1, 2 are states of Turing machines and H is Halting State.

(i) Mathematically; Turing Machine can be described using 7 tuples(Q, ∑, Γ,. δ, q_0, □, F) where

(ii) Q is the set of states, not including the halt state.

(iii) ∑ is the input alphabet that is subset of tape alphabet not including the blank symbol □.

(iv) Γ is a finite set of symbols called the tape alphabet, where □∈Γ.

(v) δ is the transition function which is defined as Q ×Γ × {L, R}.δ is written as(Present state, Input symbol) = (Next state, output symbol to replace input symbol, Direction of Head) For example δ(q, a) = (q_2, b, D).

(vi) q_0 is initial state.

(vii) F ⊆ Q is the set of final states.

ASSIGNMENT

1. Which of the following properties of r.e. (recursive enumerable) set (1) is/are r.e. (recursive enumerable)?
(i) $L = \phi$
(ii) L contains atleast 5 members
(iii)L has exactly one member.
Here, Assume L as a R.E (recursive enumerable)
(a) (i)
(b) (i) and (ii)
(c) (i),(ii) and (iii)
(d) (ii) and(iii) only

2. We say that an algorithm exists for a particular problem when the language for the problem is:
(a) Context free
(b) Context sensitive
(c) Recursive
(d) Recursively enumerable

3. Which of the following is more powerful than single tape TM?
(a) Multi-tape TM
(b) TM with multiple tracks
(c) Non – deterministic TM
(d) None of these

4. Consider the Turing Machine M;
$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$ and δ is defined by
$\delta(q_0, a) = (q_1, a, R)$
$\delta(q_1, b) = (q_2, b, R)$
$\delta(q_2\ a) = (q_2\ a, R)$
$\delta(q_2, b) = (q_3, b, R)$
q_3 is the final state.
The language accepted by Turing machine is,
(a) aba*
(b) aba^*ab
(c) aba^*b
(d) a^*ba

5. Consider the Turing machine M defined by
$M=(Q, \Sigma, \Gamma, \delta, q_0, B, F)$; And δ is defined by
$\delta(q_0, a) = (q_1, a, R)$
$\delta(q_0, b) = (q_1, b, R)$
$\delta(q_0, B) = (q_1, B,R)$
$\delta(q_1,a) = (q_0, a, L)$
$\delta(q_1, b) = (q_0, b, L)$
$\delta(q_1, B) = (q_0, B, L)$
On input ab machine will
(a) Halt in accepting state
(b) Will go into infinite loop
(c) Crash
(d) None of these

6. Let X be defined as follows:
X: Given (M), an encoding of a Turing machine. Does M halt on all inputs. Which of the following is true?
(a) X is decidable
(b) X is un-decidable but partially decidable
(c) X is un-decidable and not even partially decidable
(d) X is not a decidable problem

7. Which of the following is false?
(a) PCP is undecidable
(b) For a CFG, G it is undecidable whether L(G) is regular
(c) For two CFGs, G_1 and G_2,$L(G_1)\cap L(G_2)$ is un-decidable
(d) Given an r.e. set L, it is partially decidable whether L is regular

8. Consider the following statements:
S1: Whether given turing machine accept empty language is undecidable
S2: The complement of recursive language is recursive enumerable.
Which of the above statement is false?
(a) S1 only
(b) S2 only
(c) Both S1 and S2
(d) None of these

9. Which of the following has a read only tape?
(a) Multi-Tape TM
(b) Offline TM
(c) Multi-track TM
(d) None of these

GATE QUESTIONS

1. Let A and B be finite alphabets and let # be a symbol outside both A and B .Let f be a total function from A* to B* .We say f is computable if there exists a Turing Machine M which given an input x in A *, always halts with f(x) on its tape. Let L_f denote the language {x# f(x)|x∈A*} Which of the following statements is true?

[GATE - 2017]

(a)f is computable if and only if L_f is recursive
(b)f is computable if and only if L_f is recursively enumerable
(c)If f 'is computable then L_f is recursive, but not conversely
(d)If f is computable then L_f is recursively, enumerable, but not conversely

2. Consider the following languages
$L_1 = \{a^p | p \text{ is a prime number}\}$
$L_2\{a^nb^mc^{2m} | n \geq 0, m \geq 0\}$
$L_3 = \{a^nb^nc^{2n} | n \geq 0$
$L_4 = \{a^nb^n \mid n \geq 1\}$
Which of the following are Correct?
I. L_1 is context – free but not regular
II. L_2 is not context – free.
III. L_3 is not context – free but recursive
IV. L_4 is deterministic context – free

[GATE - 2017]

(a) I, II and IV only (b) II and III only
(c) I and IV only (d) III and IV only

3. Let L(R) be the language represented by regular expression R.L:t L(G) be the language generated by a context free grammar G. let (M) be the language accepted by a Turing Machine M. Which of the following decision problems are undecidable?
I. Given a regular expression R and a string w , is w ∈ L(R) ?
II. Given a context – free grammar G, is L(G) = ϕ?
III. Given a context – free grammar G is L(G) = Σ* for some alphabet Σ?
IV .Given a Turing machine M and a string w , is w ∈ L (M) ?

[GATE - 2017]

(a) I and IV only (b) II and III only
(c) II, III and IV only (d) III and IV only

4. Consider the following languages.
L_1 = {<M> | M takes at least 2016 steps on some input},
L_2 = {<M> | M takes at least 2016 steps on all inputs}and
L_3 = {<M> | M accepts ε},
Where for each Turing machine M, <M> denotes a specific encoding of M.
Which one of the following is **TRUE**?

[GATE - 2016]

(a) L_1 is recursive and L_2, L_3 are not recursive
(b) L_2 is recursive and L_1, L_3 are not recursive
(c) L_1, L_2 are recursive and L_3 is not recursive
(d) L_1, L_2, L_3 are recursive

5. L_1 is a recursively enumerable language over Σ. An algorithm A effectively enumerates its words as w_1, w_2, w_3, ….. Define another language L_2 over Σ∪ {#} as $\{w_i \# w_j : w_i , w_j \in L_1, i < j\}$. Here # is a new symbol. Consider the following assertions.
S_1: L_1 is recursive implies L_2 is recursive
S_2 : L_2 is recursive implies L_1 is recursive
Which of the following statements is true?

[GATE - 2004]

(a) Both S_1 and S_2 are true
(b) S_1 is true but S_2 is not necessarily true
(c) S_2 is true but S_1 is not necessarily true
(d) Neither is necessarily true

6. Define languages L_0 and L_1 as follows
L_0 = {<M, w, 0> | M halts on w}
L_1 = {<M, w, 1> | M does not halts on w}
Here <M, w, i> is a triplet, whose first component. M is an encoding of a Turing Machine, second component, w, is a string, and third component, t, is a bit.

CHAPTER - 4
DECIDABILITY AND UNDECIDABILITY

4.1 INTRODUCTION
1. Decision Problem is problem that gives answer or output in terms of Yes or No.
2. Decision problem that gives answer in terms of Yes or No based on any algorithm is called decidable.
3. Decision Problems which can have answer Yes for some time or no for sometimes are called undecidable
4. A Problem is said to be decidable if its language is recursive or it has solution or answer or Algorithm.

4.2 DECISION PROBLEM ABOUT REGULAR LANGUAGES
Some decidable Problems for finite state automaton, Regular grammar and regular languages
1. Does FA accept language?
2. Is the power of NFA and DFA same?
3. L_1 and L_2 are two regular languages. Are they closed under the following :
(i) Concatenation (ii) Intersection
(iii) Complement (iv) Transpose
(v) Kleen closure (positive transitive closure)
4. For given FA M and string w over alphabet Σ, is $w \in L(M)$?
5. For a given FA M is $L(M) = \phi$?
6. For a given FA M and alphabet Σ, is $L(M) = \Sigma^*$?
7. For a given FA M_1, and M_2, $L(M_1), L(M_2) \in \Sigma^*$ is $L(M_1) = L(M_2)$?
8. For given two regular languages L_1, L_2 over some alphabet Σ is $L_1 \subset L_2$?

4.3 DECISION PROBLEMS ABOUT CFLS AND CFGS

4.3.1 Some of the Decidable Problems
1. If L_1 and L_2 are two CFLs over some alphabets Σ then $L_1 \cup L_2$ is CFL.
2. If L_1 and L_2 are two CFLs over alphabet Σ, then L_1L_2 is CFL.
3. If L is a CFL over some alphabet Σ, then L^* is a CFL.
4. If L_1 is a regular language, L_2 is a CFL over some alphabet Σ, then $L_1 \cap L_2$ is CFL.
5. If L_1 is a regular language, L_2 is a CFL over some alphabet Σ then $L_1 \cap L_2$ is CFL.
6. For a given CFG G is $L(G) = \phi$ or not?
7. For a given CFG G, finding whether L(G) is finite or not, is decidable?
8. For given CFG G and a string w over Σ checking whether $w \in L(G)$ or not is decidable.

4.3.2 Some of the Undecidable Problems about CFGs and CFLs
1. For two given CFLs L_1 and L_2, whether $L_1 \cap L_2$ is CFL or not, is undecidable.
2. For a given CFL L over some alphabets Σ whether complement of L is CFL or not, is undecidable.
3. For a given CFG G is ambiguous
4. For two arbitrary CFGs G_1 and G_2 deciding $L(G_1) \cap L(G_2) = \phi$
5. For two arbitrary CFGs G_1 and G_2, $L(G_1) \subseteq L(G_2)$

ASSIGNMENT

1. Let $L \subseteq \Sigma^*$ where $\Sigma = \{m,n\}$, then which of the following is false?
(a) $L = \{m^c n^d | c \geq 1, d \geq 1\}$ is regular
(b) L = {x| There are more m than n} is not regular
(c) $L = \{m^a n^a | a \geq 1\}$ is regular
(d) L = {x|x has an equal number of m's and n's} is not regular

2. Let $\Sigma = \{a, b\}$, $L = \Sigma^*$ and $R = \{a^n b^n$ such that $n > 0\}$. Then the languages LUR and R are respectively,
(a) Regular, Regular
(b) Not regular, Regular
(c) Regular, not Regular
(d) Not regular, Not regular

3. Consider the following languages:
(i) $\{0^{2n} \mid n \geq 1\}$
(ii) $\{0^m 1^n 0^{m+n} \mid m \geq 1 \text{ and } n \geq 1\}$
Which of the above languages is/are regular?
(a) None (b) (i) only
(c) (ii) only (d) Both

4. Consider the following languages:
(i) $\{0^n | n$ is a prime$\}$
(ii) The set of all strings that do not have 3 consecutive 0's
Which of the above languages is/are regular sets?
(a) None (b) (i) only
(c) (ii) only (d) Both

5. Which of the following is false?
(a) Regular sets are closed under complementation
(b) Regular sets are closed under intersection
(c) Regular sets are closed under reversal
(d) None of these

6. Which of the following is false?
(i) Regular sets are closed under substitution
(ii) Regular sets are closed under homomorphism
(iii) Regular sets are closed under inverse homomorphism
(iv) Regular sets are closed under quotient with non-regular sets
(a) (iv) only
(b) (iii) & (iv) only
(c) (iii) only
(d) None of these

7. Which of the following languages is/are regular?
(i) $\{a^2, a^5, a^8, \ldots\}$
(ii) $\{0^n 1^m | \gcd(m,n)=1\}$
(iii) $\{a^n b^m | 0 \leq n \leq m\}$
(a) (i) & (iii) (b) (ii) & (iii)
(c) (i) only (d) None of these

8. Consider the following statements:
(i) Every subset of a regular language is regular
(ii) Every regular language has a regular proper subset
Choose the correct option
(a) Both (i) and (ii) are true
(b) (i) is true, (ii) is false
(c) (ii) is true, (i) is false
(d) Both are false

9. Consider the following language
(i) $L1 = L^*$ where L is any subset of Σ^* where $\Sigma = \{0\}$
(ii) $L2 = \{a_2 a_1 a_4 a_3 a_6 a_5 \ldots a_{2n} a_{2n-1} | a_1 a_2 a_3 \ldots a_{2n}, a_{2n}$ is in L, L is regular
Which of the following is ture?
(a) Both L1 and L2 are regular
(b) Only L1 is regular
(c) Only L2 is regular
(d) None of them are regular

10. Which of the following problems is undecidable?

GATE QUESTIONS

1. Which of the following decision problems are undecidable?
I. Given NFAs N_1 and N_2, is $L(N1) \cap L(N2) = \phi$
II. Given a CFG G = (N, ∑, P, S) and a string x ε ∑*, does x ε L(G)
III. Given CFGs G_1 and G_2 is $L(G_1) = L(G_2)$
IV. Given a TM M, is $L(M) = \phi$
[GATE - 2016]
(a) I and IV only (b) II and III only
(c) III and IV only (d) II and IV only

2. For any two languages L_1 and L_2 such that L_1 is context free and L_2 is recursively enumerable but not recursive, which of the following is/are necessarily true
1. $\overline{L}_1$ (complement of L_1) is recursive
2. $\overline{L}_2$ (complement of L_2) is recursive
3. $\overline{L}_1$ is context free
4. $\overline{L}_2 \cup L_2$ is recursively enumerable
[GATE - 2015]
(a) 1 only (b) 3 only
(c) 3 and 4 only (d) 1 and 4 only

3. Which of the following languages is/are regular?
L_1: $\{wxw^R | w, x \in \{a, b\}^*$ and $|w|, |x| > 0\}$, w^R is the reverse of string w
L_2: $\{a^nb^m | m \neq n$ and $m, n \geq 0\}$
L_3: $\{a^pb^qc^r | p, q, r \geq 0\}$
[GATE - 2015]
(a) L1 and L3 only (b) L2 only
(c) L2 and L3 only (d) L3 only

4. Which one of the following is TRUE?
[GATE - 2014]
(a) The language $L = \{a^nb^n | n \geq 0\}$ is regular.
(b) The language $L = \{a^n | n$ is prime$\}$ is regular.
(c) The language L = {W | w has 3k + 1b's for some $k \in N$ with $\Sigma = \{a, b\}$} is regular.
(d) The language $L = \{ww | w \in \Sigma^*$ with $\Sigma = \{0, 1\}\}$ is regular.

5. Let L be a language and $\overline{L}$ be its complement. Which one of the following is NOT a viable possibility?
[GATE - 2014]
(a) Neither L nor $\overline{L}$ is recursively enumerable (r.e.).
(b) One of L and $\overline{L}$ is R.E but not recursive; the other is not R.E.
(c) Both L and $\overline{L}$ are R.E but not recursive.
(d) Both L and $\overline{L}$ are recursive.

6. If L_1 $\{a^n | n \geq 0\}$ and $L_2 = \{ b^n | n \geq 0\}$, consider
I. $L_1.L_2$ is a regular language
II. $L_1.L_2 = \{a^nb^n | n \geq 0\}$
Which one of the following is CORRECT?
[GATE - 2014]
(a) Only I
(b) Only II
(c) Both I and II
(d) Neither I nor II

7. Let $A \leq_m B$ denotes that language A is mapping reducible (also known as many-to-one reducible) to language B. which one of the following is FALSE?
[GATE - 2014]
(a) If $A \leq_m B$ and B is recursive then A is recursive.
(b) If $A \leq_m B$ and A is undecidable then B is undecidable.
(c) If $A \leq_m B$ and B is recursively enumerable then A is recursively enumerable.
(d) If $A \leq_m B$ and B is not recursively enumerable then A is not recursively enumerable.

8. Let < M > be the encoding of a Turing machine as a string over Σ = {0, 1}. Let L = {<

CHAPTER - 5
P, NP, NP-HARD AND NP-COMPLETE PROBLEMS

5.1 INTRODUCTION

1. There are many problems exist in the world. Some of the problems are very easy and some are difficult. Easy problems are also called solvable and difficult problems are those problems which are not solvable or take more time to solve.
2. Solvable problems are called tractable problems.

5.2 ABSTRACT PROBLEM

1. It is defined as binary relation on a set I of problem instances and a set S of problem solutions.
2. Abstract decision problem is a function that maps the instance set I to the solution set {0, 1}.

For example, decision problem is related to shortest-path is the Problem path.
i = < G, u, v, k> is the instance of the shortest path problem that belongs to set I of shortest path. If path (i) = yes, it implies there is a path from u to v has almost k edges. Otherwise path (i) = No.

5.3 ENCODING PART

1.It is a mapping of abstract objects from a set to the set of binary strings such as set N = {0, 1, 2, 3, 4, …} ⇒ e (30 = 11.
2.Similarly are abstract objects such as polygons, graphs, functions, ordered pairs, programs can be encoded as binary strings.
3.Encoding also exists in shortest part abstract decision problem where every instance from set S can be encoded
4.It transforms abstract problem to concrete problem.
5.The computer algorithm that solves abstract decision problem actually takes on encoding of a problem instance as input.
6.Concrete problem has input instances as a binary strings.
7.Polynomial-time solvability of a problem also depends upon encoding but it is assumed that it is independent of encoding procedure.
8.Theory of computation discipline allows us to express the relation between decision problems and algorithms that solve them concisely.
9.If there is an abstract decision problem with instance set I, its encoding set e(I) and solution set S = {0, 1}. Then, if an algorithm/machine model accepts a string $x \in e(I)$ if I given as input then language (L) of machine/Algorithm will be L ={ $x \in e(I)$: S(x) = 1 }. So, it includes all accepted strings but it rejects $x \in e(I)$ and S(x) = 0
10.Language L/problems is said to be decidable if every binary string in L is accepted by machine/algorithm and every binary string into in L is rejected by the machine/algorithm. Therefore, all Turing machine problems/languages are decidable.
11.A language L is said to be decided in polynomial time, if there is an algorithm for which a constant k exist and for strings of any-length n $x \in \{0, 1\}^*$, the algorithm correctly decides whether $x \in L$ in time O (n^k).
12.Turing machine languages are decided in finite amount of time. It also implies that they are decidable
13.Some algorithm/machine accepts all $x \in L$, but loop forever. If $x \notin L$. These languages are called recursive enumerable.

ASSIGNMENT

1. P $\neq$ NP
(a) True (b) False
(c) Can't say (d) None of these

2. Consider the following problems:
1. Finding out in directed graph whether Hamiltonian cycle exists.
2. Given Boolean formula is 2CNF
3. Finding out shortest path
Find out which is correct?
(a) All three are NP complete problem
(b) (2) and (3) are NP complete (1) is NP Hard
(c) (1) is NP Complete,(2).and (3) can be solved in polynomial time
(d) All three will be solved in polynomial time

3. A problem is in NP and as hard as any problem in NP.
The given problem is:
(a) NP hard
(b) NP
(c) NP hard $\cap$ NP – complete
(d) NP complete

4. Jitendra and Shantanu have been asked to show certain problem A is NP-complete. Jitendra shows a polynomial time reduction from the clique problem to A and Shantanu shows polynomial time reduction from A to clique problem. Which of the following can be inferred from this reduction?
(a)A is NP hard but not NP complete
(b)A is in NP, but is not NP complete
(c)A is NP-complete
(d)A is neither NP hard, nor in NP

5. If a problem requires time $\Theta(n^{100})$ problem is:
(a)Tractable (b) Intractable
(c)NP-hard (d) None of these

6. NP–languages are closed under which of the following operation
I. Union
II. Intersection
III. Complement
IV. Concatenation
V. Kleene star
(a) I, II, IV, V
(b) I, II, III, IV, V
(c) I, II, III,
(d) IV, V

7. Suppose we are able to solve Hamiltonian cycle in polynomial time, then which of the following relations will hold?
(a)NP – P = ϕ (b) P $\subseteq$ NP
(c)P $\subseteq$ CO-NP (d) P = NP

8. Determine the correctness or otherwise of the following Assertion [A] and the Reason [R].
Assertion: Any given problem in P will also be in NP
Reason: P $\subset$NP
(a)Both statements are not related and invalid
(b)Both statements are not related
(c)Both statements are related and valid reason is valid
(d)Both statements are related but reason is invalid.

9. Polynomial time algorithm is closed under which of the following operation?
(i) Addition
(ii) Multiplication
(iii)Composition
(iv)Complement
(a) (i), (ii) only
(b) (i), (ii) and (iii) only
(c) All
(d) None of these

10. A polynomial time algorithm makes at most constant number of calls to polynomial time subroutines. The resulting algorithm runs in:
(a)Polynomial time
(b)Non – polynomial time

GATE QUESTIONS

1. LanguageL1 is polynomial time reducible to Language L2. Language L3 is polynomial time reducible to L2. Which is turn is polynomial time reducible to language L4.
Which of the following is/are true?
I. If $L4 \in P$, $L2 \in P$
II. If $L1 \in P$ or $L3 \in P$, then $L2 \in P$
III. If $L1 \in P$, if and only if $L3 \in P$
IV. If $L4 \in P$, then $L1 \in P$ and $L3 \in P$

[GATE - 2015]

(a) II only (b) III only
(c) I and IV only (d) I only

2. Consider two decision problems Q_1, Q_2 such that Q_1 reduces in polynomial time to 3-SAT and 3-SAT reduces in polynomial time to Q_2. Then which one of the following is consistent with the above statement?

[GATE - 2015]

(a) Q_1 is in NP, Q_2 is NP hard.
(b) Q_2 is in NP, Q_1 is NP hard.
(c) Both Q_1 and Q_2 are in NP.
(d) Both Q_1 and Q_2 are NP hard.

3. Consider the following statements.
I. The complement of every Turing decidable language is Turing decidable
II. There exists some language which is in NP but is not Turing decidable
III. If L is a language in NP, L is Turing decidable
Which of the above statements is/are true?

[GATE - 2015]

(a) Only II (b) Only III
(c) Only I and II (d) Only I and III

4. Consider the decision problem 2CNFSAT defined as follows :
{ϕ | ϕ is a satisfiable propositional formula in CNF with at most two literals per clause}
For example, $\phi = (x_1 \vee x_2) \wedge (x_1 \vee \overline{x_3}) \wedge (x_2 \vee x_4)$ is a Boolean formula and it is in 2CNFSAT. The decision problem 2CNFSAT is

[GATE - 2014]

(a) NP-Complete
(b) Solvable in polynomial time by reduction to directed graph reachability.
(c) Solvable in constant time since any input instance is satisfiable.
(d) NP-hard, but not NP-complete.

5. Which of the following statements are TRUE?
1.The problem of determining whether there exists a cycle in an undirected graph is in P.
2.The problem of determining whether there exists a cycle in an undirected graph is in NP.
3.If a problem A is NP-complete, there exists a non-deterministic polynomial time algorithm to solve A.

[GATE - 2013]

(a) 1, 2 and 3 (b) 1 and 2 only
(c) 2 and 3 only (d) 1 and 3 only

6. Assuming $P \neq NP$, which of the following is TRUE?

[GATE - 2012]

(a) NP-complete = NP
(b) NP-complete ∩ P = $\varnothing$
(c) NP-hard = NP
(d) P = NP-complete

7. Let S be an NP-complete problem Q and R be two other problems not known to be in NP. Q is polynomial-time reducible to S and S is polynomial-time reducible to R. which one of the following statements is true?

[GATE - 2006]

(a) R is NP-complete (b) R is NP-hard
(c) Q is NP-complete (d) Q is NP-hard

SECTION - B
COMPILER DESIGN

CHAPTER - 1
LEXICAL ANALYSIS

1.1 INTRODUCTION
There are various language processors that process/convert High-Level language code into Machine-level code. They can be categorized as
1. Compiler
2. Interpreter
3. Assembler

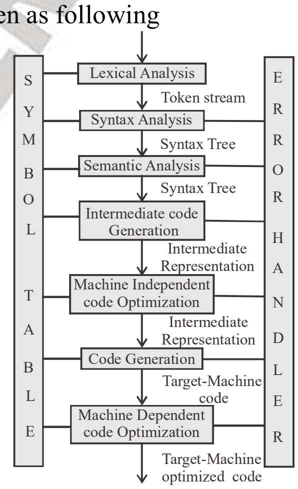
1.1.1 Compiler
1.It is a program that translates a source code in one language to machine language.
2.It is faster than an interpreter at mapping inputs to outputs.

1.1.2 Interpreter
1.It directly executes the operations specified in the source program as input supplied by the user.
2.It usually gives better error diagnostics as it executes the source program statement by statement.

> Java language Processors combine both interpreter and compiler.

1.1.3 Assembler
1. It translates source code into a language that is intermediate between High-Level language and Machine Level Language.
2. It translates source code in assembly language to relocatable machine code as its output.

1.2 STRUCTURE OF COMPILER
1. Generally, A Compiler is designed to have several phases that are responsible for the functions such as Lexical Analysis, Syntax Analysis, Semantic Analysis, Intermediate Code Generation, Code Optimization etc.
2. The structure of Compiler is given as following



ASSIGNMENT

1. Compiler time errors do not include
(a) Lexical errors (b) Syntactic errors
(c) Semantic errors (d) None of these

2. The range checking for certain values, array subscripts and case statements selectors are examples of
(a) Semantic errors (b) Dynamic errors
(c) Syntactic errors (d) None of these

3. A compiler which allows only the modified section of the source code to be recompiled is called as
(a) Incremental compiler
(b) Re-configurable compiler
(c) Dynamic compiler
(d) Subjective compiler

4. Which table is a permanent database that has an entry for each terminal symbol?
(a) Terminal table (b) Literal table
(c) Identifier table (d) Reductions

5. The task of lexical analysis phase is
(a)To parse the source program into the basic elements or tokens of the language
(b) To build a literal table and an identifier table
(c) To build a uniform symbol table
(d) All of the above

6. Consider the following statements
S_1: The set of string described by a rule is called pattern associated with the token.
S_2: A lexeme is a sequence of characters in the source program that is matched by pattern for token.
Which of above statements is are true?
(a) Both S1 and S2 are true
(b) S_1 is true S_2 is false
(c) S_2 is true S_1 is false
(d) Both S_1 and S_2 are false

7. Which of the following strings can definitely be said to be token without looking at the next input character while compiling a pascal program?
(i) Begin
(ii) Program
(iii) <>
(a) (i) (b) (ii)
(c) (iii) (d) all of the above

8. In compiler, keywords of a language are recognized during
(a) Parsing of the program
(b) Code generation
(c) Lexical analysis
(d) Dataflow analysis

9. Which of the following is used to group the characters into tokens?
(a) Parser
(b) Code optimization
(c) Code generator
(d) Scanner

10. Which of the following grammars are not phase- structured?
(a) Regular
(b) Context free grammar
(c) Context sensitive
(d) None of the above

11. Cross-compiler is a compiler
(a)That generates object code for its host machine
(b)Which is written in a language that is the same as the source language
(c)Which is written in a language that is different from the source language.
(d)That runs on one machine and produces object code for another machine

12. How many tokens are contained in the following FORTAN statement:

GATE QUESTIONS

1. Consider the following Syntax Directed Translation Scheme (SDTS), with non-terminals {S, A} and terminals {a,b}.
S → aA {print 1}
S → a {print 2}
A→ Sb {print 3}
Using the above SDTS, the output printed by a bottom-up parser, for the input **aab** is:
[GATE - 2016]
(a) 1 3 2 (b) 2 2 3
(c) 2 3 1 (d) syntax error

2. In a compiler, keywords of a language are recognized during
[GATE - 2011]
(a) Parsing of the program
(b) The code generation
(c) The lexical analysis of the program
(d) Dataflow analysis

3. Which data structure in a complier is used for managing information about variables and their attributes?
[GATE - 2010]
(a) Abstract syntax tree (b) Symbol table
(c) Semantic stack (d) Parse table

4. Consider line number 3 of the following C-program.

int main () {	/* Line 1 */
int i, n;	/* Line 2 */
fro (i = 0, i < n, i + +);	/* Line 3 */
}	

Identify the compiler's response about this line while creating the object-module
[GATE - 2005]
(a) No compilation error
(b) Only a lexical error
(c) Only syntactic errors
(d) Both lexical and syntactic errors

5. Consider a program P that consists of two source modules M_1 and M_2 contained in two different files. If M_1 contains a reference to a function defined in M_2, the reference will be resolved at
[GATE - 2004]
(a) Edit-time (b) Compile-time
(c) Link-time (d) Load-time

6. Which of the following is NOT an advantage of using shared; dynamically linked libraries as opposed to using statically linked libraries?
[GATE - 2003]
(a) Smaller sizes of executable
(b) Lesser overall page fault rate in the system
(c) Faster program startup
(d) Existing programs need not be re-linked to take advantage of newer versions of libraries

7. The number of tokens in the following C statement
Print f ("i = %d, &i = %x", i, &i); is
[GATE - 2000]
(a) 3 (b) 26
(c) 10 (d) 21

CHAPTER - 2
SYNTAX ANALYSIS

2.1 INTRODUCTION
1.It is the second phase of compilation.
2.Its purpose is to recombine, obtained tokens from Lexical Analysis and to output the structure of text.
3.The structure of the text is rejected by Tree data structure that is called here Syntax Tree of the text.
4.Tokens of Lexical Analysis are at the leaf level of the syntax Tree. When leaves are read from left to right, the sequence is the same as in the input text.
5.It is a method for recovery of common errors
6.It also reject in valid texts by reporting syntax errors
7.The syntactic structure of well formed programs, which contains functions, statement out of expressions, function out of declarations and statements etc.
8.Syntax of language constructs can be specified by context free grammars or BNF (Backus – Naur Form) notation.

2.2 ROLE OF PARSER
1.It takes a string of tokens from the lexical analyzer and verifies that the string of token names can be generated by the grammar for the source language.
2.It reports any syntax errors in the program language.
3.It constructs a parse tree for well formed programs.
4.There are three general types of parsers for grammars: Universal , top down and Bottom up
5.Commonly parsing methods used in compilers can be classified as being either top – down or bottom up

2.3 SYNTAX ERROR HANDLING
1.Syntax Analyzer handles syntactic errors such as misplaced semicolons, extra | missing braces i.e. {or}, misplaced else etc.
2.It uses two error recovery strategies having broad applicability Panic-Mode recovery, Phrase Level Recovery, Error Productions, Global correction.

2.3.1 Panic-Mode Recovery
1 In this method, on discovering an error, the parser discards input symbols one at a time until one of a designated set of synchronizing tokens (delimiters such as})
2.While correction, it often skips a considerable amount of input without checking it for additional errors.

2.3.1.1 Advantage
1.It is simple method
2.It is guaranteed not to go into an in-finite loop.

2.3.2 Phrase - Level Recovery
1. Here, p when parser detects an error, it performs local correction on the remaining input.
2. Local correction means to replace a prefix of the remaining input by same string that allow the parser to continue.

GATE QUESTIONS

1. Consider the following expression grammar G :
E → E −T|T
T→ T +F |F
F→ (E) |id
Which of the following grammars is not left recursive , but us equivalent to G ?
[GATE - 2017]
(a) E → E −T|T
T→ T +F |F
F→ (E) |id
(b) E → TE'
E'→ -TE' |ε
T → T+F |F
F→ (E) |id
(c) E → TX
X→ -TX |ε
T→ FY
Y → FY|ε
F→ (E) |id
(d) E→TX |(TX)
X→−TX|+TX|ε
T→id

2. Which of the following statements about parser is/are CORRECT ?
I. Canonical LR is more powerful than SLR
II. SLR is more powerful than LALR
III. SLR is more powerful than CLR
[GATE - 2017]
(a) I only (b) II only
(c) III only (d) II and III only

3. Consider the following grammar :
P→ xQRS
Q → yz |z
R → w|ε
S → y
What is FOLLOW (Q)?
[GATE - 2017]
(a) {R} (b) {w}
(c) {w, y} (d) {w, $}

4. A student wrote two context-free grammars **G1** and **G2** for generating a single C-like array declaration. The dimension of the array is at least one. For example,
int a[10][3];
The grammars use D as the start symbol, and use six terminal symbols int; id[ ] num.

Grammar G1	**Grammar G2**
D → int L;	D → int L;
L → id[E	L → id E
E → num]	E → E[num]
E → num][E	E → [num]

Which of the grammars correctly generate the declaration mentioned above?
[GATE - 2016]
(a) Both G1 and G2
(b) Only G1
(c) Only G2
(d) Neither G1 nor G2

5. Which one of the following grammars is free from left recursion?
[GATE - 2016]
(a) S → AB
A → Aa | b
B → c
(b) S → Ab | Bb | c
A → Bd | ε
B → e
(c) S → Aa | B
A → Bb | Sc | ε
B → d
(d) S → Aa | Bb | c
A → Bd | ε
B → Ae | ε

6. Consider the grammar defined by the following production rules, with two operators * and +
S → T ∗ P

CHAPTER - 3
SEMANTIC ANALYSIS

3.1INTRODUCTION
1.Generally any string/statement derived from production in a grammar specifies the required programming constructs of the language that are called semantic rules.
2.Syntax-Directed Definition is termed for attaching rules or program fragments to productions in a grammar.
3.Before Syntax–Directed Translation, Syntax-directed Definition is done.
4.Generally, Syntax-Directed Translation is to construct a parse tree and then to compute the values of attributes at the nodes of the tree according to the syntax-Directed Definitions (SDD).

3.2 SYNTAX-DIRECTED DEFINITION (SDD)
1.It is a context-free grammar together with attributes and rules.
Attributes are associated with grammar symbols and rules are associated with the productions.
2.Attributes can be of any kind: numbers, types, table reference or strings.
3.If X is a symbol and a is one of its attributes, then X.a denotes the value of a at a particular parse-tree node labeled X.

Example.
If we define the semantic rules to be associated with each production of the grammar. Then, we call its Syntax-Directed Definition. It is follows as.

Productions of Grammar	Associated Semantic Rule
$E \rightarrow E_1$	$E.val = E_1.\ val$
$E_1 \rightarrow E_2 + T$	$E_1.val = E_2.\ val + T.\ val$
$E_1 \rightarrow T$	E_1 val = T. val
$T \rightarrow T_1 * F$	$T.\ val = T_1 .\ val * F.\ val$
$T \rightarrow F$	T. val = F. val
$F \rightarrow (E_1)$	$F.\ val = E_1.\ val$
$F \rightarrow$ digit	F. val = digit

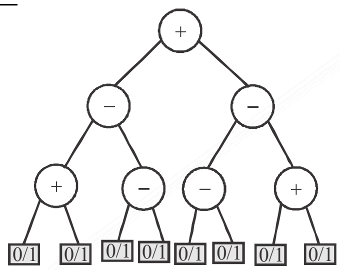
In above grammar, each non-terminal has a single attribute called val.
Let us take semantic rule, $T.\ Val = T_1.\ Val \times F.\ Val$ that computes value of head T by multiplying the values of head T_1 and head F. Similarly, we can understand all other semantic rules.
There are two kinds of attributes for non-terminals
1. Synthesized Attributed   2. Inherited Attribute

1. Synthesized Attributed
1.It defines a non-terminal at any node of parse tree.
2.It is defined by semantic rule associated with the production at any node.
3.Synthesized attribute at node N is defined in terms of attribute values at the children of the node N and at N itself.

2. Inherited Attribute
1.It also defines any non-terminal at a parse tree node.

GATE QUESTIONS

1. Consider the expression tree shown. Each leaf represents a numerical value, which can either be 0 or 1. Over all possible choices of the values at the leaves, the maximum possible value of the expression represented by the tree is ______.



[GATE - 2014]

2. Consider the following translation scheme.
S → ER
R → *E {print ('*'); R | ε
E → F + E {print ('+'); | F
F → (S) | id {print (id. Value);}
Here id is a token that represents an integer and id. value represents the corresponding integer value. For an input "2* 3 + 4" this translation scheme prints

[GATE - 2006]

(a) 2 * 3 + 4 (b) 2 * + 3 4
(c) 2 3 * 4 + (d) 2 3 4 + *

Common Data for Q. 3 & Q. 4
Consider the following expression grammar. The semantic rules for expression calculation are stated next to each grammar production.
E → number E.val = number.val
| E '+' $E^{(1)}$.val = $E^{(2)}$.val + $E^{(3)}$ .val
| E '×' E $E^{(1)}$ .val = $E^{(2)}$. Val × $E^{(3)}$ .val;

3. The above grammar and the semantic rules are fed to a YACC tool (which is an LALR(1) parser generator) for parsing and evaluating arithmetic expressions. Which one of the following is true about the action of YACC for the given grammar?

[GATE - 2005]

(a)It detects recursion and eliminates recursion
(b)It detects reduce-reduce conflict, and resolves
(c)It detects shift-reduce conflict, and resolves the conflict in favor of a shift over a reduce action
(d)It detects shift-reduce conflict and resolves the conflict in favor of a reduce over a shift action

4. Assume the conflicts in Part (a) of this question are resolved and an LALR(1) parser is generated for parsing arithmetic expressions as per the given grammar. Consider an expression 3 × 2 + 1. What precedence and associativity properties does the generated parser realize?

[GATE - 2005]

(a) Equal precedence and left associativity; expression is evaluated to 7
(b) Equal precedence and right associativiy; expression is evaluated to 9
(c) Precedence of '×' is higher than that of '+', and both operators are left associative; expression is evaluated to 7
(d) Precedence of '+' is higher than that of '×', and both operators are left associative; expression is evaluated to 9

5. Consider the grammar with the following translation rules and E as the start symbol.
E → E_1 # T {E.value = E_1.value* T.value}
| T {E.value = T. value}
T → T_1 & F {T.value = T_1. Value + F.Value}
| F {T.value = F. value}
F → num {F. value = num. value}
Compute E.value for the root of the parse tree for the expression: 2 # 3 & 5 # 6 & 4.

[GATE - 2004]

(a) 200 (b) 180
(c) 160 (d) 40

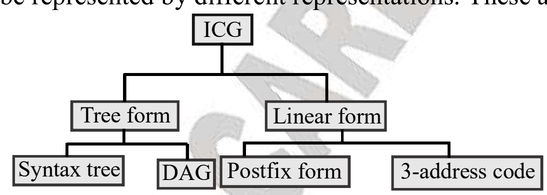
CHAPTER - 4
INTERMEDIATE CODE GENERATION

4.1 INTRODUCTION
Intermediate code generation using the parse rule produces a language from input language. In compiler the front end translates a source program into an intermediate code, from which back end generates target code. Details of languages are included in back end as far as possible.

Why we need Intermediate Code?
Intermediate code has property that it is simple enough to be translated to assembly code.

The benefits of using machine independent intermediate form
1.Retargeting is facilitated
2.Machine independent code optimizer can be applied to intermediate representation.

4.2 REPRESENTATION OF INTERMEDIATE CODE GENERATION
Intermediate code can be represented by different representations. These are classified as follows



4.2.1 Postfix Notation
Postfix Notation is written with operator after operands in the expression.
e.g.:- infix way of writing sum of a and b is a + b and postfix notation of same infix expression is ab+. In general if E_1 and E_2 are any postfix expression and r is any binary operator, the result of applying r to E_1 and E_2 is indicated as E_1E_2 r. No parentheses are needed in postfix notation because the position and number of arguments of the operators only one way to decode a postfix expression.
Example. If infix expression is (a – b) ∗ (c + d) + (a – b) then its postfix notation
ab – cd + ∗ ab - +

4.2.2 Syntax Tree
Syntax tree is condensed form of parse tree. The operator and keywords nodes of parse tree are moved to their parent and chain of single productions is replaced by single link.
Example.
Syntax tree of following infix expression
(a) (a + b) ∗ (a + b + c)

CHAPTER - 5
CODE OPTIMIZATION

5.1 INTRODUCTION
1.Code optimization is a set of methods of code modification to improve code quality and efficiency. A program may be optimized so that it becomes smaller in size to consume less memory and or performs fewer input/output operations to execute more rapidly.
2.Optimization can be performed by automatic optimizers or programmers. An optimizer software tool or built-in unit of compiler (so called optimized compiler). Modern processes can also optimize the execution of code instruction.
3.Code optimization involves complex analysis of intermediate code and performance of various transformations but every optimizing transformation must also preserve the semantics of program when attempting an optimizing transformation. The following criteria should be applied.
(i) Optimization should capture most of the potential improvement without an unreasonable amount of effort.
(ii)The optimization should be such that the meaning of source program is preserved.
(iii) Optimization should, on average, reduce the time and space expanded by the object code.
(iv) Optimization can be machine dependent or machine independent.
(v) Machine dependent optimization requires knowledge of target machine while machine independent optimization can be performed independently of the target machine for which compiler is generating codes.

5.2 ELIMINATION OF COMMON SUB EXPRESSION
An occurrence of expression E is called a common sub expression if E was previously computed, and the values of variable in E have not changed since the previous computation. We can avoid re-computing the expression if we can use previously computed value.
Example.
If execution order of statements is following
1. t_6:=4× I 2. X:a[t_6] 3. t_7:=4×i 4. t_8:=4×j 5. t_9:=a[t_8]
6. a[t_4]:t_9 7. t_{10}:4×j 8. A[t_{10}]:x can be written
1. t_6:4×I 2. X:a[t_6] 3. t_8:4× j 4. t_9:=a[t_8] 5. a[t_6]:=t_9 6. a[t_8]:X
Here t_7 eliminated by using t_6 and t_{10} is eliminated by using t_8 instead of t_{10}.

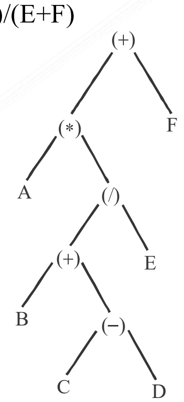
5.3 METHODS OF CODE OPTIMIZATION
There are various methods by which we can optimize any code.
1.loop optimization
2.Strength Reduction
3.Constant folding
4.Redundancy elimination
5.Dead code elimination
6.Algebraic expression

5.3.1 Loop Optimization
As we know the statement executed inside the loop is the number of times the loop runs. Due to these loops, a program spends the bulk of time. So to decrease the running time, There is need to

ASSIGNMENT

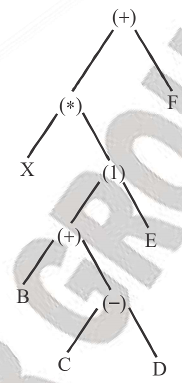
1. Peephole optimization is a form of
(a) Loop optimization
(b) Constant folding
(c) Local optimization
(d) None of these

2. Which of the following expression is represented by the parse tree given below?
(a) A+B*C-D/E+F
(b) A+(B+(C-D))/E+F
(c) A+B*C-D/(E+F)
(d) A+B*(C-D)/(E+F)



3. Which of the following is/are incorrect about intermediate code representation?
(i) The indirect triples and quadruples are required about the same amount of space.
(ii) The indirect triples are much efficient for recording of code as compared to quadruples
(a) (i) only (b) (ii) only
(c) (ii) and (i) (d) neither (i) nor (ii)

4. Which of the following expressions is represented by the parse tree given below?



(a) A+B*C-D/E+F
(b) A*(B+(C-D)) /E +F
(c) A+B*C-D/ (E+F)
(d) A+B*(C-D) / (E+F)

5. The method which merges the bodies of two loops is
(a) Loop unrolling (b) Loop ramming
(c) Constant folding (d) None of these

6. Loop is a collection of nodes that is
(a) Strongly connected
(b) Loosely connected and has a unique entry
(c) Strongly connected and has a unique entry
(d) None of these

7. The identification of common sub-expression and replacement of run-time computations by compile-time computations is
(a) Local optimization
(b) Loop optimization
(c) Constant folding
(d) Data flow analysis

8. The specific tasks storage manager performs are
(a) Allocation/deallocation of storage to programs
(b) Protection of storage area allocated to a program form illegal access by other programs in the system.

GATE QUESTIONS

1. Match the following:

List-I

P. lexical analysis
Q. Top down parsing
R. Semantic Analysis
S. Runtime environments

List-II

(i) Leftmost derivation
(ii) Type checking
(iii) Regular expressions
(iv) Activation records

[GATE - 2016]

(a) P-i , Q-ii, R-iv, S-iii
(b) P-iii, Q-i, R-ii, S-iv
(c) P-ii, Q-iii, R- i, S-iv
(d) P-iv, Q-i, R-ii, S-iii

2. Consider the following code segment.

x = u – t;
y = x * v;
x = y + w;
y = t – z;
y = x * y;

The minimum number of total variables required to convert the above code segment to static single assignment form is ________.

[GATE - 2016]

3. Consider the basic block given below.

a = b + c , c = a + d
d = b + c , e = d – b
a = e + b

The minimum number of nodes and edges present in the DAG representation of the above basic block respectively are

[GATE - 2014]

(a) 6 and 6 (b) 8 and 10
(c) 9 and 12 (d) 4 and 4

4. Which one of the following is FALSE?

[GATE - 2014]

(a) A basic block is a sequence of instructions where control enters the sequence at the beginning and exists at the end.
(b) Available expression analysis can be used for common sub expression elimination.
(c) Live variable analysis can be used for dead code elimination.
(d) $x = 4 \times 5 \Rightarrow x$ is an example of common sub expression elimination.

Common Data for Q. 5 & Q. 6

The following code segment is executed on a processor which allows only register operands in its instructions. Each instruction can have at most two source operands and one destinations operand. Assume that all variables are dead after this code segment.

```
c = a + b;
d = c * a;
e = c + a;
x = c * c;
if (x > a) {
y = a * a;
}
Else {
d = d * d;
e = e * e;
}
```

5. Suppose the instruction set architecture of the processor has only two registers. The only allowed complier optimization is code motion, which moves statements from one place to another while preserving correctness. What is the minimum number of spills to memory in the compiled code?

[GATE - 2013]

(a) 0 (b) 1
(c) 2 (d) 3

6. What is the minimum number of registers needed in the instruction set architecture of the processor to compile this code segment without any spill to memory? Do not apply any optimization other than optimizing register allocation?