VidyaJyothi Institute of Technology (Autonomous)

(Accredited by NAAC & NBA, Approved By A.I.C.T.E., New Delhi, Permanently Affiliated to JNTU, Hyderabad)
(Aziz Nagar, C.B.Post, Hyderabad -500075)

Department of Information Technology

Course File

Regulations	: R20
Batch	:2020-2024
Academic Year	:2021-2022
Program	:B.Tech. (IT)
Course Name	: Data Structures (DS)
Year / Sem.	:II/I
Course Code	:A43504
Pre-requisites	: Programing for Problem Solving- I &II
Course Coordinator	: Mrs. K. Shireesha

Head of the Department
Dept. of Information Technology
Vidya Jyothi Institute of Technology
Aziz Nagar Gate, C.B. Post
Hyderabad - 500 075, Telangana

Index

INDEX

S.NO.	ITEM DESCRIPTION
1	Course Information Sheet
2	Syllabus
3	Text Books, Reference Books, Web/Internet Resources
4	Time table
5	Program Educational Objectives(PEOS) and Program Outcomes(POs)
6	Program Specific Outcomes(PSOs)
7	Course Outcomes(COs)
8	Mapping of Course Outcomes, POs and PSOs
9	Course Schedule
10	Lecture Plan/Teaching Plan
11	Unit wise Date of Completion and Remarks
12	Assignment Questions
13	Unit wise Question Bank
14	Mid Question Papers
15	End Exam papers
16	Content Beyond Syllabus
17	Unit wise PPTs and lecture notes
18	CO Attainment - Direct and Indirect
19	Course end survey form

Syllabus

Syllabus

Unit – I	**Data Structures:** Introduction, Types of data structures, Static and Dynamic representation of data structure and comparison. **Stacks:** Stacks definition, operations on stacks, Representation and evaluation of expressions using Infix, Prefix and Postfix, Algorithms for conversions and evaluations of expressions from infix to prefix and postfix using stack. **Queues:** Types of Queues- Circular Queue, Deque and operations.
Unit – II	**Trees:** Basic terminologies, Types of Binary Tree: Complete and Full Binary Tree, Extended Binary Trees, Representation of Trees using Arrays and Linked lists (advantages and disadvantages), Tree Traversal, Representation of Algebraic expressions, Threaded Binary Trees.
Unit – III	**Advanced concepts on trees:** Representation and Creation of Binary Search Trees (BST), Operations on BST, Representation and advantages of AVL Trees, algorithms & operations on AVL Trees, Multi-way trees, Definition and advantages of B-trees, B+ Trees, Red-Black Trees.
Unit – IV	**Graphs**: Basic terminology, Representation of graphs: sequential representation, Adjacency, Path Matrix) Linked representation. Graph Traversals-Breadth First Search, Depth First Search algorithms. Spanning Tree, Minimum Spanning Trees- Prim's Algorithm, Kruskals Algorithm, Dijkstra Algorithm.
Unit – V	**Hashing:** General Idea, Hash Functions, collisions, Collision avoidance techniques, Separate Chaining ,Open Addressing-Linear probing, Quadratic Probing, Double Hashing, Rehashing, Extensible Hashing, Implementation of Dictionaries

Text Books

&

ReferenceBooks

Text Books,Reference Books/web sources

Text Books:
1.Data Structures Using C, 2^{nd} Edition Reema Thereja OXFORD higherEducation
2.Fundamentals of Data Structures, 2^{nd}Horowitz and Sahani, *Galgotia Publications* Pvt Ltd Delhi India.
Reference Books:
1.Data Structures, Seymour Lipschutz, Schaum's Outlines, Tata McGraw-Hill, Special Second Edition.
2.Data Structures Using C and C++‖, Aaron M. Tenenbaum, YedidyahLangsam and Moshe J. Augenstein PHI Learning Private Limited, DelhiIndia.
3.Data Structures, A Pseudo code Approach with C, Richard F.Gillberg& Behrouz A. Forouzan, Cengage Learning, India Edition, Second Edition,2005
Other Resources:
Vjit.ac.in/it/study-material/

Program Educational Objectives(PEOs)& Program Outcomes(POs)

Vidya Jyothi Institute of Technology
(Affiliated to JNTUH)
AziznagarGate,C.B.Post,Hyderabad-500 075
DEPARTMENT OF INFORMATION TECHNOLOGY

Program Educational Objectives

(PEOs)

PEO1: Core Capabilities / Competence: Impart profound knowledge in humanities and basic sciences along with core engineering concepts for practical understanding and project development.

PEO2: Career Advancement: Enrich analytical and industry based technical skills through ICT for accomplishing research, higher education and entrepreneurship

PEO3: Life-Long Learning: Infuse life-long learning, professional ethics, adaptation to innovation and effective communication skills with a sense of social awareness.

Programme Outcomes (PO's)

1. **Engineering Knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Vidya Jyothi Institute of Technology

(Affiliated to JNTUH)

AziznagarGate,C.B.Post,Hyderabad-500 075

DEPARTMENT OF INFORMATION TECHNOLOGY

Program Specific Outcomes

(PSOs)

PSO1: Enhanced ability in applying mathematical abstractions and algorithmic design along with programming tools to solve complexities involved in efficient programming.

PSO2: Developed effective software skills and documentation ability for graduates to become employable/ higher studies/ Entrepreneur/ Researcher.

Course Outcomes (Cos)

Course Outcomes (CO's)

Course Name: C204 (DATA STRUCTURES)

After completing this course the student must demonstrate the knowledge and ability to	
CO1	Understand the concepts of Stacks and Queues with their applications.
CO2	Analyze various operations on Binary trees.
CO3	Examine of various concepts of binary trees with real time applications.
CO4	Analyze the shortest path algorithm on graph data structures.
CO5	Outline the concepts of hashing, collision and its resolution methods using hash functions.

Mapping of Course Outcomes, POs and PSOs

Course Schedule

Course Schedule:

Distribution of Hours in Unit-Wise

Unit	Topics	Reference section	Total No.Of.Hours
I	Introduction, Types of data structures, Static and Dynamic representation of data structure and comparison. **Stacks:** Stacks definition, operations on stacks, Representation and evaluation of expressions using Infix, Prefix and Postfix, Algorithms for conversions and evaluations of expressions from infix to prefix and postfix using stack. **Queues:** Types of Queues- Circular Queue, Deque and operations.	Text Book1	**10**
II	**Trees:** Basic terminologies, Types of Binary Tree: Complete and Full Binary Tree, Extended Binary Trees, Representation of Trees using Arrays and Linked lists (advantages and disadvantages), Tree Traversal, Representation of Algebraic expressions, Threaded Binary Trees.	Text Book2	**12**
III	**Advanced concepts on trees:** Representation and Creation of Binary Search Trees (BST), Operations on BST, Representation and advantages of AVL Trees, algorithms & operations on AVL Trees, Multi-way trees, Definition and advantages of B-trees, B+ Trees, Red-Black Trees.	Text Book2	**10**
IV	**Graphs**: Basic terminology, Representation of graphs: sequential representation, Adjacency, Path Matrix) Linked representation. Graph Traversals-Breadth First Search, Depth First Search algorithms. Spanning Tree, Minimum Spanning Trees- Prim's Algorithm, Kruskals Algorithm, Dijkstra Algorithm.	Text Book2	**10**
V	**Hashing**: General Idea, Hash Functions, collisions, Collision avoidance techniques, Separate Chaining ,Open Addressing-Linear probing, Quadratic Probing, Double Hashing, Rehashing, Extensible Hashing, Implementation of Dictionaries	Reference book3	**12**
	Total No.of classes		54
	Tutorial classes:2 per unit Assignments:2(before every mid1 and mid2 examinations)		

TB1: Data Structures Using C, 2nd Edition Reema Thereja OXFORD higher Education
TB2: Fundamentals of Data Structures, 2nd Horowitz and Sahani, *Galgotia Publications* Pvt Ltd Delhi India.
RB3: Data Structures, A Pseudo code Approach with C, Richard F.Gillberg & Behrouz A. Forouzan, Cengage Learning, India Edition, Second Edition, 2005.

Lecture Plan /Teaching Plan

S. No.	Topic	Expected Date of Completion	Actual Date of Completion	Teaching Learning Process	Co's	Evaluation	Textbooks /references
Unit-I (CO1)							
1	Introduction, Types of data structures,	20/9/2021	20/9		CO1	MID-1	TB1
2	Static and Dynamic representation of data structure and comparison.	21/9/2021	21/9		CO1	MID-1	TB1
3	**Stacks:** Stacks definition, operations on stacks,	22/9/2021	22/9	PPT	CO1	MID-1	TB1
4	Representation and evaluation of expressions using Infix	23/9/2021	23/9		CO1	MID-1	TB1
5	Prefix and Postfix, Algorithms for conversions	24/9/2021	24/9		CO1	MID-1	TB1
6	Examples solved on infix to prefix conversions	25/9/2021	25/9	FLIPPED CLASS ROOM	CO1	MID-1	TB1
7	evaluations of expressions from infix to prefix	28/9/2021	28/9		CO1	MID-1	TB1
8	Postfix evaluation using stack.	1/10/2021	01/10		CO1	MID-1	TB1
9	**Queues:** Types of Queues-Circular Queue	2/10/2021	02/10		CO1	MID-1	TB1
10	Deque and operations.	4/10/2021	4/10		CO1	MID-1	TB1
UNIT II(CO2)							
1	**Trees:** Basic terminologies,	5/10/2021	5/10		CO2	MID-1	TB2
2	Types of Binary Tree: Complete Binary Tree	6/10/2021	6/10		CO2	MID-1	TB2
3	Full Binary Tree	8/10/2021	8/10		CO2	MID-1	TB2
4	Extended Binary Trees	9/10/2021	9/10		CO2	MID-1	TB2
5	Representation of Trees using Arrays	11/10/2021	10/10		CO2	MID-1	TB2
6	Representation of Trees using Arrays with example	12/10/2021	11/10 12/10		CO2	MID-1	TB2
7	Representation of Trees using Linked lists	15/10/2021	15/10	PPT	CO2	MID-1	TB2
8	Representation of Trees using Linked lists with example	16/10/2021	16/10		CO2	MID-1	TB2
9	advantages and disadvantages	20/10/2021	20/10		CO2	MID-1	TB2

TB1: Data Structures Using C, 2nd Edition Reema Thereja OXFORD higher Education
TB2: Fundamentals of Data Structures, 2nd Horowitz and Sahani, *Galgotia Publications* Pvt Ltd Delhi India.
RB3: Data Structures, A Pseudo code Approach with C, Richard F.Gillberg & Behrouz A. Forouzan, Cengage Learning, India Edition, Second Edition, 2005.

10	Tree Traversal with examples	22/10/2021	22/10	FLIPPED CLASS ROOM	CO2	MID-1	TB2
11	Representation of Algebraic expressions,	27/10/2021	27/10		CO2	MID-1	TB2
12	Threaded Binary Trees.	30/10/2021	30/10		CO2	MID-1	TB2
UNI- III (CO3)							
1	**Advanced concepts on trees:** Representation	3/11/2021	3/11		CO3	MID-1	TB2
2	Creation of Binary Search Trees (BST)	5/11/2021	5/11		CO3	MID-1	TB2
3	Operations on BST	6/11/2021	6/11		CO3	MID-1	TB2
4	Representation and advantages of AVL Trees	9/11/2021	9/11	PPT	CO3	MID-1	TB2
5	Algorithms & operations on AVL Trees	19/11/2021	19/11		CO3	MID-2	TB2
6	Multi-way trees.	20/11/2021	20/11		CO3	MID-2	TB2
7	Definition and advantages of B-trees	22/11/2021	22/11		CO3	MID-2	TB2
8	B+ Trees	23/11/2021	23/11		CO3	MID-2	TB2
9	,Red-Black Trees	26/11/2021	26/11		CO3	MID-2	TB2
10	Examples on AVL tree construction	27/11/2021	27/11	FLIPPED CLASS ROOM	CO3	MID-2	TB2
UNIT-IV (CO4)							
1	**Graphs**: Basic terminology	29/11/2021	29/11		CO4	MID-2	TB2
2	Representation of graphs: sequential representation, Adjacency, Path Matrix)	30/11/2021	30/11		CO4	MID-2	TB2
3	Adjacency Matrix and List	31/11/2021	31/11		CO4	MID-2	TB2
4	Linked representation.	1/12/2021	01/12	PPT	CO4	MID-2	
5	Graph Traversals-Breadth First Search	3/12/2021	3/12		CO4	MID-2	TB2
6	Depth First Search algorithms.	4/12/2021	4/12		CO4	MID-2	TB2
7	Spanning Tree, Minimum Spanning Trees	5/12/2021	5/12		CO4	MID-2	TB2
8	- Prim's Algorithm	14/12/2019	14/12		CO4	MID-2	TB2
9	Kruskals Algorithm, Dijkstra Algorithm	15/12/2021	15/12		CO4	MID-2	TB2

TB1: Data Structures Using C, 2nd Edition Reema Thereja OXFORD higher Education
TB2: Fundamentals of Data Structures, 2nd Horowitz and Sahani, *Galgotia Publications* Pvt Ltd Delhi India.
RB3:Data Structures, A Pseudo code Approach with C, Richard F.Gillberg & Behrouz A. Forouzan, Cengage Learning, India Edition, Second Edition, 2005.

10	Examples on graph traversals	16/12/2021	16/12	FLIPPED CLASS ROOM	CO4	MID-2	TB2
UNIT-V (CO5)							
1	**Hashing:** General Idea, Hash Functions	18/12/2021	18/12		CO5	MID-2	RB3
2	Hash Functions	20/12/2021	20/12		CO5	MID-2	RB3
3	Collisions, Collision avoidance techniques	22/12/2021	22/12	PPT	CO5	MID-2	RB3
4	Separate Chaining	23/12/2021	23/12		CO5	MID-2	RB3
5	Open Addressing-Linear probing, Difference between open hashing and closed hashing	24/12/2021	24/12		CO5	MID-2	RB3
6	Linear probing with example	27/12/2021	27/12		CO5	MID-2	RB3
7	Quadratic Probing with examples	29/12/2021	29/12		CO5	MID-2	RB3
8	Double Hashing	30/12/2021	30/12		CO5	MID-2	RB3
9	Double Hashing with example	31/12/2021	31/12		CO5	MID-2	RB3
10	Rehashing with examples	4/01/2022	04/01		CO5	MID-2	RB3
11	Extensible Hashing	5/01/2022	05/01		CO5	MID-2	RB3
12	Implementation of Dictionaries	6/01/2022	06/01		CO5	MID-2	RB3
13	RIVISION	7/01/2022	07/01	FLIPPED CLASS ROOM	C05	MID-2	RB3
14	RIVISION	08/01/2022	08/01	FLIPPED CLASS ROOM	CO5	MID-2	RB3
Total No. of classes:56							

TB1: Data Structures Using C, 2nd Edition Reema Thereja OXFORD higher Education
TB2: Fundamentals of Data Structures, 2nd Horowitz and Sahani, *Galgotia Publications* Pvt Ltd Delhi India.
RB3: Data Structures, A Pseudo code Approach with C, Richard F.Gillberg & Behrouz A. Forouzan, Cengage Learning, India Edition, Second Edition, 2005.

Unit wise Date of Completion and Remarks

Date of Unit completion & Remarks

Unit – I	
Date:	4/10/2021
Remarks:	Completed in time
Unit – II	
Date:	30/10/2021
Remarks:	Completed in time
Unit – III	
Date:	27/11/2021
Remarks:	Completed in time
Unit – IV	
Date:	16/12/2021
Remarks:	Completed in time
Unit – V	
Date:	8/01/2022
Remarks:	Completed in time

Unit wise Assignment Questions

Assignment 1

Unit-1	
1	Convert the given expression A+B/(C*D)/E^F-(G*H) into prefix ,post fix notation. (Level-2,CO1)
2	Perform insertion operation on a circular queue 10,15,23,45,67,78,identify rear and front pointer positions after all insertions. (Level-3,CO1)
Unit-2	
1	Construct a binary search tree with the following values 25,37,4,8,90,23,56,44,6,12 (Level-6,CO2)
2	Construct a AVL tree with the values 20,30,10,45,67,78,23,45,56,67 (Level-6,CO2)
Unit-3	
1	Compare the different type of trees. (Level-3,CO3)

Name: P. Sai Kumar
Roll No: 20911A12A9
Branch & sec: IT-B
Subject: DS

Assignment-1

1. Convert the given expression A+B/(C*D)/E^F-(G*H) into prefix, postfix notation.

5/5

Sol:- Given,
Infix expression A+B/(C*D)/E^F-(G*H)

Infix to Postfix:-

Input Expression	Stack	Postfix.
A	NILL	A
+	+	A
B	+	AB
/	+/	AB
(	+/(	AB
C	+/C	ABC
*	+/(*	ABC
D	+/(*	ABCD
)	+/()	ABCD*
/	+/	ABCD*/
E	+/	ABCD*/E
^	+/^	ABCD*/E
F	+/^	ABCD*/EF
-	+/-	ABCD*/EF^
(	+/-(	ABCD*/EF^
G	+/-(	ABCD*/EF^G
*	+/-(*	ABCD*/EF^G
H	+/-(*	ABCD*/EF^GH
)	+/-( )	ABCD*/EF^GH*

Postfix Expression:- ABCD*/EF^GH*-/+

Infix to Prefix:-

(H*G)-F^E/(D*C)/B+A

Input Expression	Stack	Postfix.
(	(	NILL
H	(	H
*	(*	H
G	(*	HG
)	( )	HG*
-	-	HG*
F	-	HG*F
^	-^	HG*F
E	-^	HG*FE
/	-/	HG*FE^
(	-/(	HG*FE^
D	-/(	HG*FE^D
*	-/(*	HG*FE^D
C	-/(*	HG*FE^DC
)	-/( )	HG*FE^DC*
/	-/	HG*FE^DC*/
B	-/	HG*FE^DC*/B
+	-+	HG*FE^DC*/B/
A	-+	HG*FE^DC*/B/A

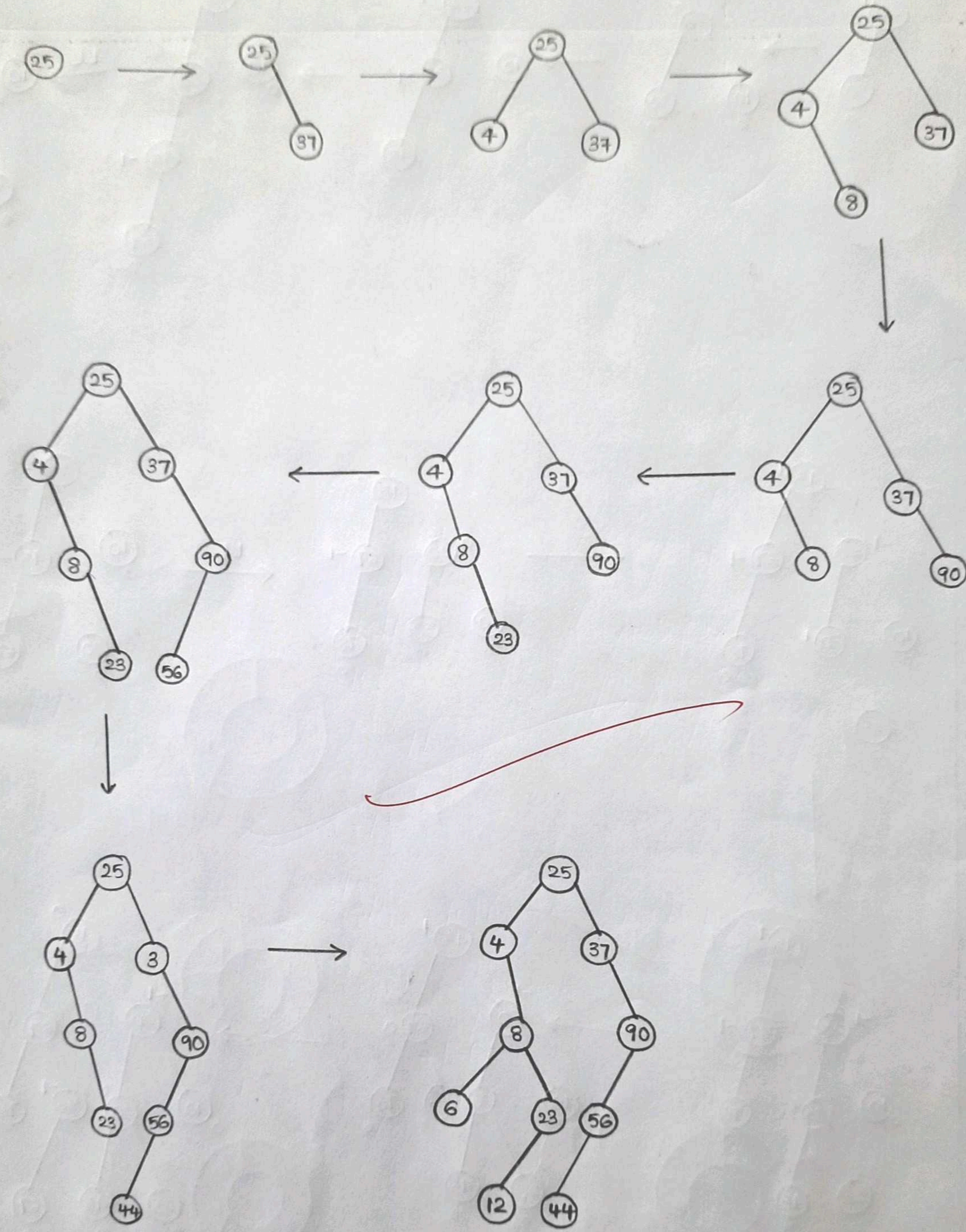
HG*FE^DC*/B/A+-

Prefix Expression: -+A/B/*CD^EF*GH

3. Construct a binary search tree with the following values.

25, 37, 4, 8, 90, 23, 56, 44, 6, 12.

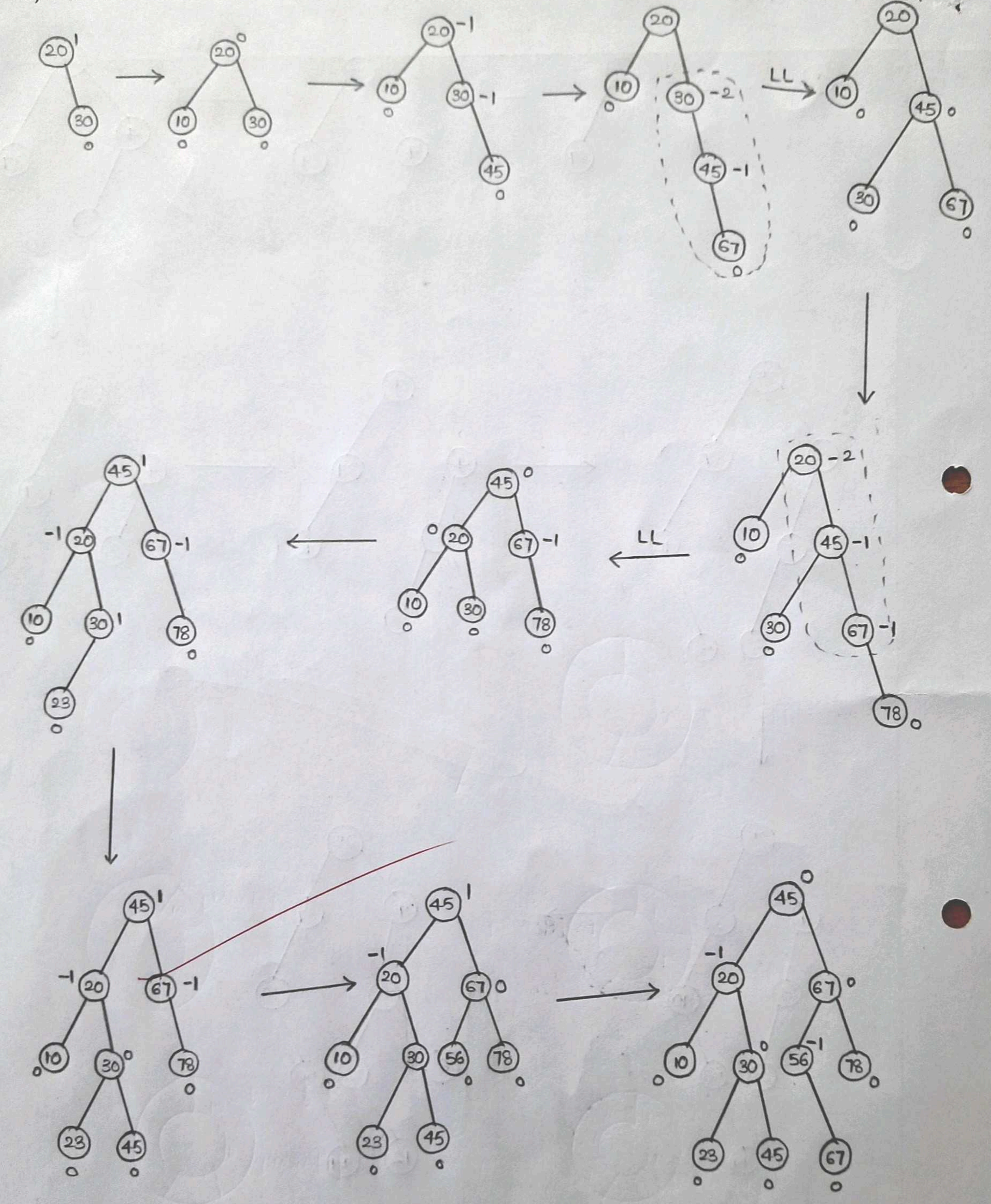
Ans:-



4. Construct a AVL tree with the values

20, 30, 10, 45, 67, 78, 23, 45, 56, 67.

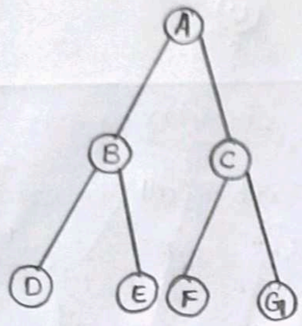
Ans:-



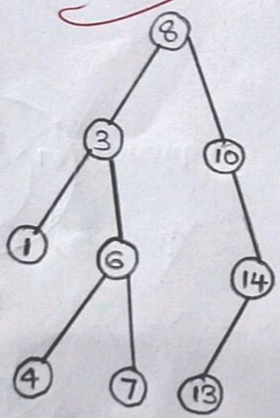
5. Compare the different types of trees.

Ans:- 1. General Tree:- If no constraint is placed on the tree's hierarchy, a tree is called a general tree. Every node may have infinite number of children in General Tree. The tree is the super-set of all other trees.

2. Binary Tree:- The binary tree is the kind of tree in which most of two children can be found for each parent. The children are known as the left child and right child. This is more popular than most other trees.

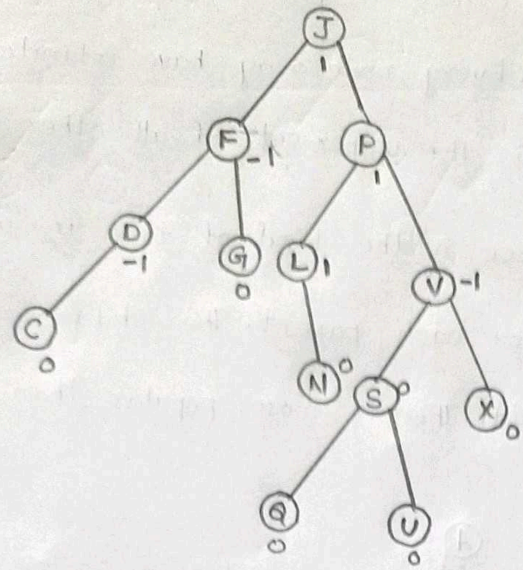


3. Binary Search Tree:- Binary Search Tree (BST) is a binary tree extension with several optional restrictions. The left child value of a node should in BST be less than or equal to the parent value, and right child value should always be greater than or equal to the parent's value.



4. AVL Tree:- AVL tree is a binary search tree self balancing. On behalf of the inventors Adelson-Velshi and Landis, the name AVL is given. This was the first tree that balanced dynamically. A balancing factor is allocated for each node in the AVL tree, based on whether the tree is balanced or not. The hight of the child nodes is atmost 1. In the AVL tree, the correct

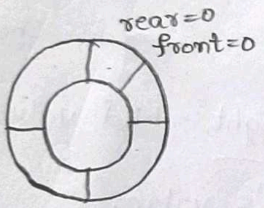
balance factor is 1, 0, -1.

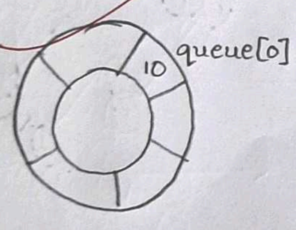


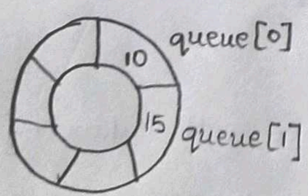
2. Perform insertion operation on a circular queue 10, 15, 23, 45, 67, 78, identify rear and front pointer positions after all insertions.

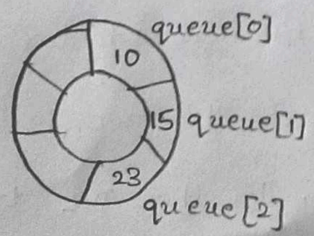
Ans: Given elements 10, 15, 23, 45, 67, 78.

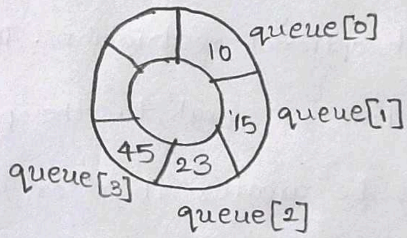
Before insertion rear=0 ; front=0

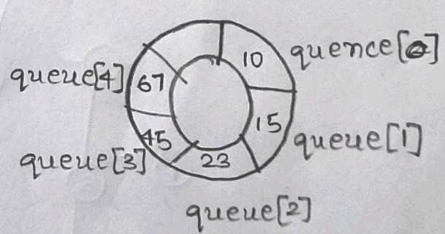


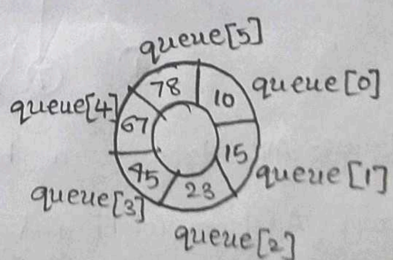








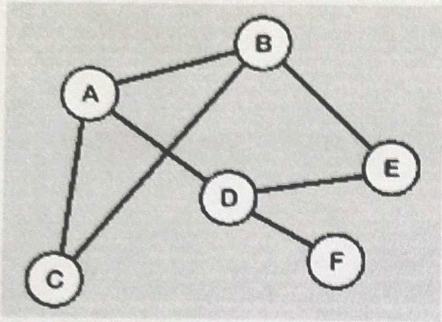




rear position = queue[0]

front position = queue[5]

Assignment 2

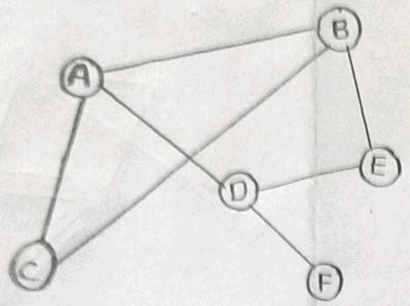
Unit-3	
1	Explain DFS graphs traversal algorithms and Illustrate DFS traversals of following graph  (Level-3,CO3)
2	Explain DIJKSTRA Algorithm with example (Level-4,CO3)
Unit-4	
1	Analyze input (371, 323, 173, 199, 344, 679, 989) and hash function h(x)=x mod 10, Show the result using Separate Chaining. (Level-4,CO4)
2	Define Minimum Spanning tree? Explain prims's Algorithm with example. (Level-2,3,CO4)
Unit-5	
1	Analyze input (371, 323, 173, 199, 344, 679, 989) and hash function h(x)=x mod 10, Show the result using quadratic probing. (Level-4,CO5)

Name :- P. Sai Kumar
Roll No :- 20911A12A9
Branch & Sec :- IT-B
Subject :- DS.

Assignment - 2.

5/5

1. Explain DFS graphs traversal algorithms and Illustrate DFS traversals of the following graph.



Ans:- * DFS traversal of a graph, produces a spanning tree as final result.

* Spanning tree is a graph without any loops.

* We use Stack data structure with maximum size of total number of vertices in the graph to implement DFS traversal.

We use the following steps to implement DFS traversal.

Step-1 :- Define a Stack of size total number of vertices in the graph.

Step-2 :- Select any vertex as starting point for traversal. Visit the vertex and push it on to the Stack.

Step-3 :- Visit any of the adjacent vertex of the vertex which is at the top of the Stack which is not visited and push it on to the stack.

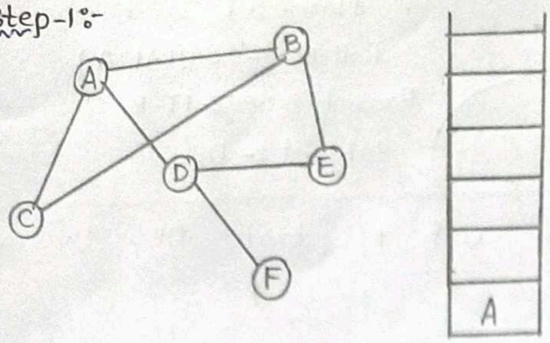
Step-4 :- Repeat Step-3 until there are no new vertex to be visit from the vertex on top of the Stack.

Step-5 :- When there is no new vertex to be visit then use back tracking and pop one vertex from the stack.

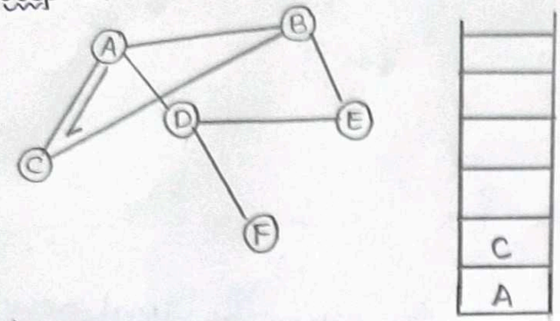
Step-6 :- Repeat Step 3, 4 and 5 until stack becomes Empty.

Step-7 :- When stack becomes Empty, then produce final spanning tree by removing unused edges from the graph.

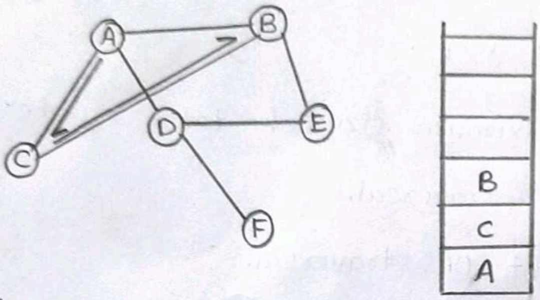
Step-1:-



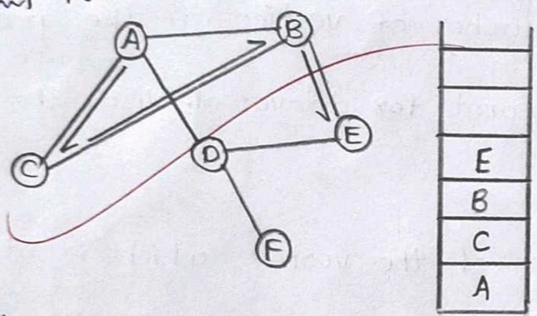
Step-2:



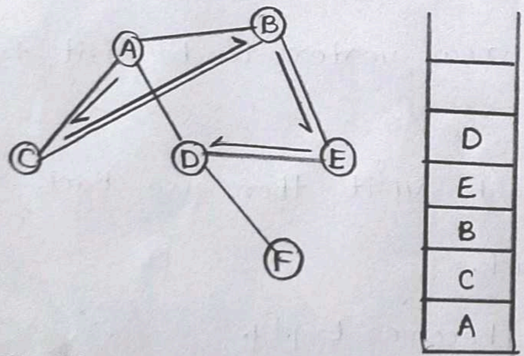
Step-3:-



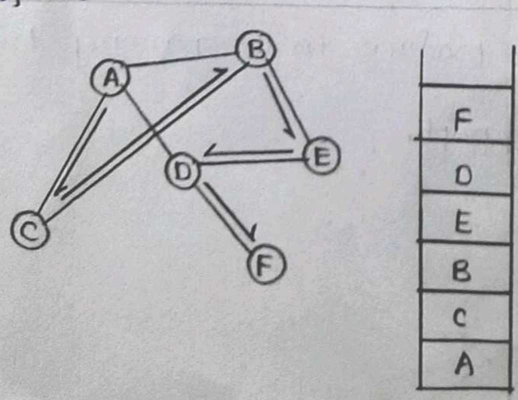
Step-4:-



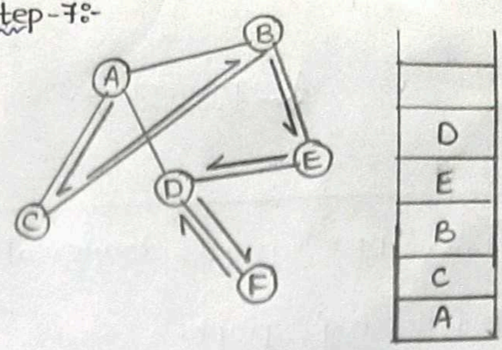
Step-5:-



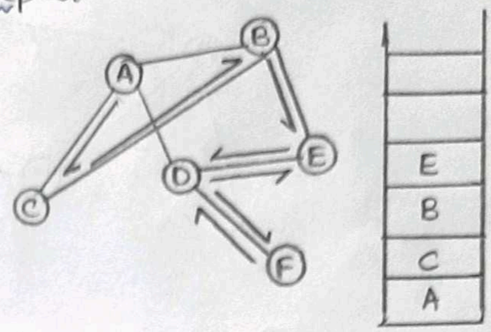
Step-6:-



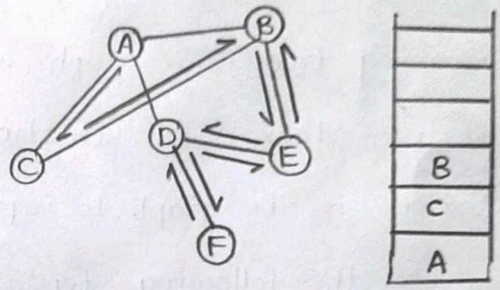
Step-7:-



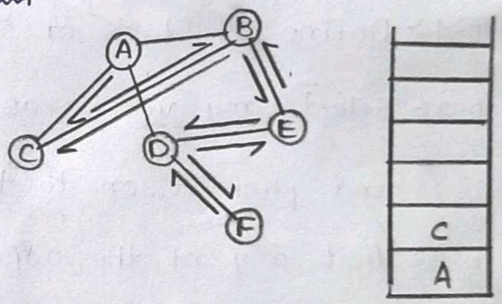
Step-8:-



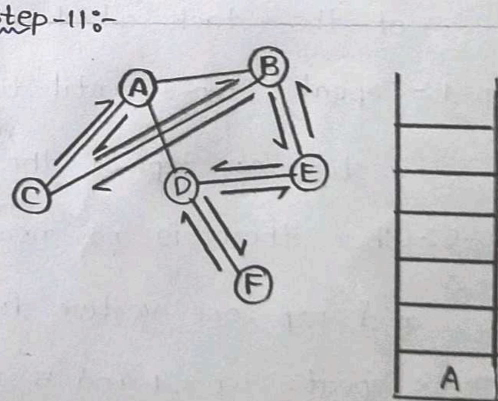
Step-9:-



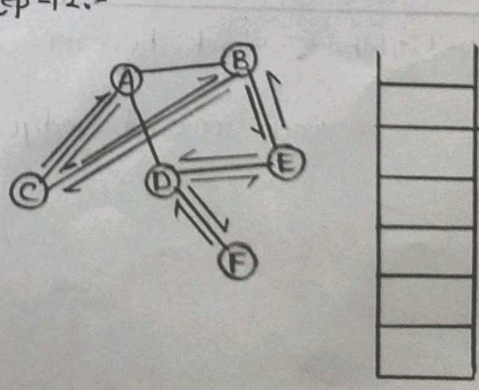
Step-10:-



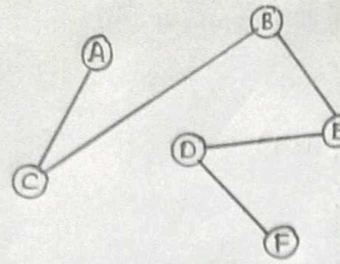
Step-11:-



Step-12:-



Spanning Tree:-



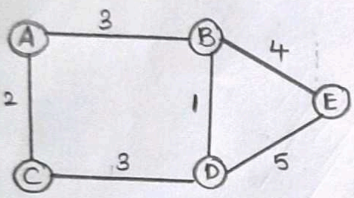
2. Explain DIJKSTRA Algorithm with example.

Ans:- DIJKSTRA Algorithm:-

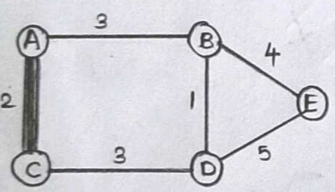
Dijkstra's algorithm is also known as the shortest path algorithm. It is an algorithm used to find the shortest path between nodes of the graph. The algorithm creates the tree of the shortest paths from the starting source vertex from all other points in the graph.

Example:-

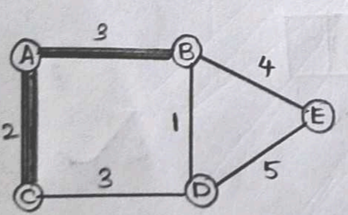
Step-1:-



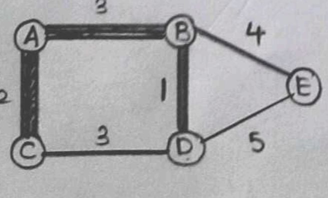
Step-2:-



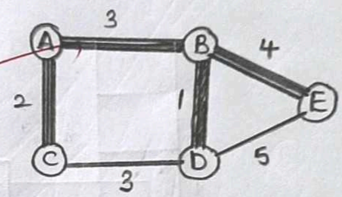
Step-3:-



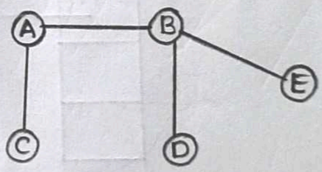
Step-4:-



Step-5:-



Spanning Tree:-



3. Analyze input (371, 323, 173, 199, 344, 679, 989) and hash function $h(x) = x \bmod 10$. Show the result in Seperate Chaining.

Ans:- Given,

input (371, 323, 173, 199, 344, 679, 989)

hash function $h(x) = x \bmod 10$.

Now,

$h(371) = 371 \bmod 10 = 1$

$h(323) = 323 \bmod 10 = 3$

$h(173) = 173 \bmod 10 = 3$

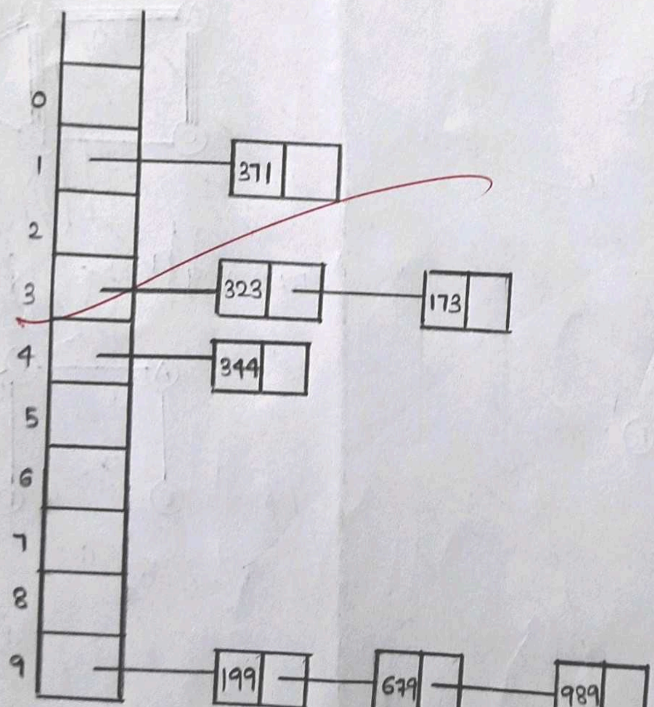
$h(199) = 199 \bmod 10 = 9$

$h(344) = 344 \bmod 10 = 4$

$h(679) = 679 \bmod 10 = 9$

$h(989) = 989 \bmod 10 = 9$

Representation using seperate Chaining:-



4. Define Minimum Spanning Tree? Explain Prim's Algorithm with Example.

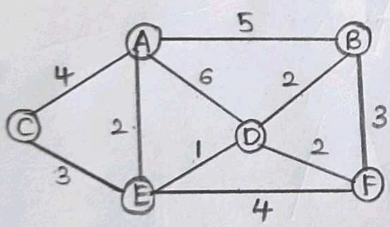
Ans:- **Minimum Spanning Tree:-** The minimum spanning tree for a given graph is the spanning tree of minimum cost for the graph.

Prim's Algorithm:-

Prim's Algorithm finds the minimum cost spanning tree by selecting the edges one by one as follows:

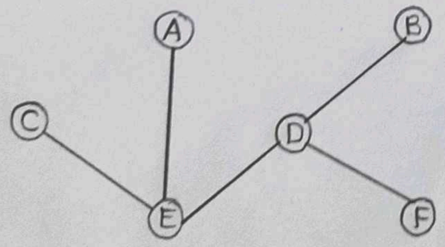
1. All vertices are marked as not visited.
2. Any vertex v you like is chosen as starting vertex and is marked as visited.
3. The smallest-weighted edge e = (u,v) which connects one vertex v inside the cluster C with another vertex u outside of C, is chosen and is added to the MST.
4. The process is repeated until a spanning tree is formed.

Example:-



ED	1
AE	2
DF	2
DB	2
CE	3
BF	3
AC	4
EF	4
AB	5
AD	6

Minimum Spanning Tree:-



5. Analyze input (371, 323, 173, 199, 344, 679, 989) and hash function $h(x) = x \bmod 10$. Show the result using quadratic probing.

Ans:- Given,

input (371, 323, 173, 199, 344, 679, 989)

hash function $h(x) = x \bmod 10$.

$\rightarrow h(371) = 371 \bmod 10 = 1$

$\rightarrow h(323) = 323 \bmod 10 = 3$

$\rightarrow h(173) = 173 \bmod 10 = 3$

$\rightarrow h(173) = (173 + 1^2) \bmod 10$

$= (174) \bmod 10$

$= (4)$

$h(173) = (173 + 2^2) \bmod 10$

$= (177) \bmod 10$

$= 7$

$\rightarrow h(199) = 199 \bmod 10$

$= 9$

$\rightarrow h(344) = 344 \bmod 10 = 4$

$\rightarrow h(679) = 679 \bmod 10 = 9$

$h(679) = (679 + 1^2) \bmod 10$

$= (680) \bmod 10$

$= 0$

$\rightarrow h(989) = (989) \bmod 10$

$= 9$

$h(989) = (989 + 1^2) \bmod 10$

$= (990) \bmod 10$

$= 0$

$h(989) = (989 + 2^2) \bmod 10$

$= (993) \bmod 10$

$= 3$

$h(989) = (989 + 3^2) \bmod 10$

$= (998) \bmod 10$

$= 8$

Representation using quadratic probing.

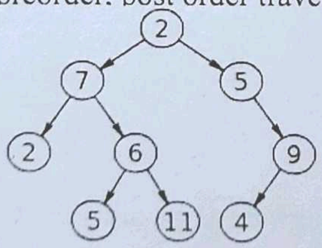
0	679
1	371
2	
3	323
4	344
5	
6	
7	173
8	989
9	

Unit wise Question Bank

Unit Wise Questions

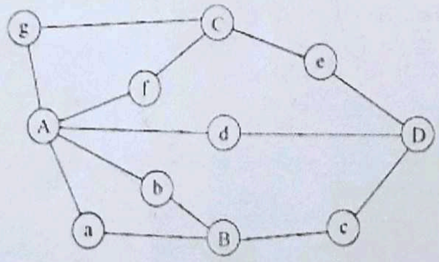
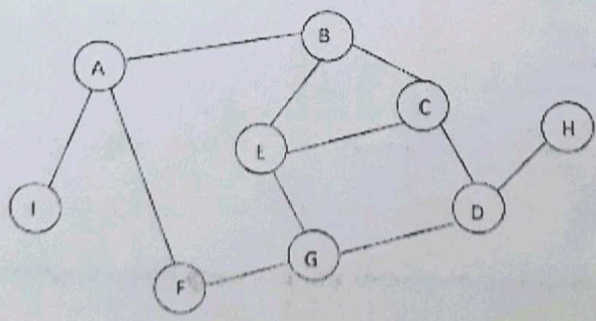
Question	Blooms Level
UNIT-1	
Short Answer Questions	
1. Define data structure.	1
2. List linear and nonlinear data structures.	1
3. List the operations performed in the Linear Data Structure.	1
4. Define abstract data type (ADT).	1
5. Define string.	1
6. List various string handling functions.	1
7. Define strcmp ().	1
8. Write a C-Program to check whether the given string is palindrome or not by using string handling functions.	3
9. Define Stack.	1
10. Define the term top?	1
11. Explain the over flow condition on stack?	2
12. Explain the term Underflow on stack?	2
13. List the applications of stack.	1
14. What is recursion?	1
15. Define prefix and postfix.	1
16. State the rules to be followed during infix to postfix conversions.	1
17. Convert the infix expression (a+b)-(c*d) into post fix form.	2
18. List how Stacks are represented in data structure.	1
19. Discuss which data structure used in recursion.	2
20. Explain the difference between stack implementation using array and linked list.	2
21. Write the necessity of infix to post fix conversion.	1
Long Answer Questions	
1. Explain the various string handling functions with example.	3

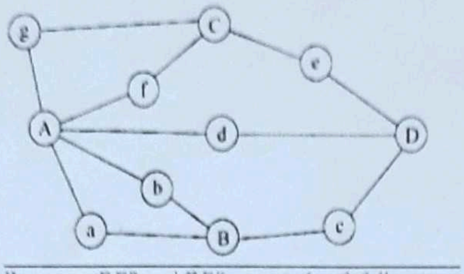
2. Define a pointer. Write a C-program to reverse a string by using pointers.	
3. Write an algorithm for basic operations on Stack.	3
4. Explain the procedure to evaluate postfix expression.	2
5. Evaluate the following postfix expression: 6 2 3 + - 3 8 2 / + * 2 | 3 +.	5
6. Explain the procedure to convert infix expression into postfix Expression.	3
7. Convert the following expression A+(B*C)-((D*E+F)/G) into post fix form.	2
8. Convert the expression ((A + B) * C - (D - E) * (F + G)) into post fix form.	2
9. Transform the following expression to postfix expression using stacks. (a+b)*((d-e)+f).	2
10. Convert infix expression into its equivalent post fix expression A*(B+D)/E-F*(G+H/K).	2
11. Transform the following expression to postfix expression using stacks. (A+B)*(C/(D-E)+F)- G.	2
12. Write C programs to implement stack ADT using Arrays	3
13. Write C programs to implement stack ADT using Linked list.	3
UNIT-2	
Short Answer Questions	
1. Define Tree.	1
2. What are the properties of tree?	1
3. List the applications of Trees	1
4. Define the terms node, degree, siblings, depth/height, level.	1
5. Define the term descendent of a tree.	1
6. Define path in a tree.	1
7. Define an empty tree.	1
8. Define Binary Tree	1
9. Explain the properties of a binary tree.	2
10. What is the difference between a tree and a binary tree.	1
11. Define full binary tree	1
12. Define complete binary tree	1
13. Define balanced binary tree,	1
14. Define a right-skewed binary tree and Left-skewed binary tree.	1

15. State the properties of a Binary Tree.	11
16. Discuss how to represent Binary Tree	2
17. Define threaded binary tree.	1
18. Explain the rules to construct threaded binary tree.	2
19. What are the advantages and disadvantages of sequential (or) static (or) array representation on a binary tree?	3
20. What are the advantages and disadvantages of linked representation a binary tree.	3
21. Define an Expression tree.	1
22. Draw an expression tree for the following expression A*(B+D)/E-F*(G+H/K).	3
23. List the different traversing techniques of a binary tree.	1
Long Answer Questions	
1. Explain Binary tree ADT.	2
2. Discuss representation of binary tree.	2
3. Explain tree traversals with example.	2
4. Define and explain the following.	2
a. Complete binary tree.	
b. Full binary tree.	
c. Expression tree.	
5. Define threaded binary tree? Explain the impact of such a representation on the tree traversal procedure?	3
6. Define an expression tree for the following expression ((A + B) * C - (D - E) * (F + G)) and write the different tree traversals for above expression.	3
7. Write in order, preorder, post order traversal of the following tree. 	3
8. Given In order traversal of a binary tree is D,G,B,E,A,H,F,I,C and pre order traversal is A,B,D,G,E,C,F,H,I construct binary tree.	3
UNIT-3	

Short Answer Questions	1
1. Define balanced search tree	1
2. Define binary search tree with example.	1
3. State the operations on binary search tree.	1
4. What are the drawbacks of a binary search tree?	2
5. What is the difference between a binary tree and a binary search tree?	2
6. Define balance factor and what the balance factor value of avl tree is.	1
7. Define AVL tree with example.	1
8. List the different AVL tree rotations to insert a node.	1
9. Explain the L-R rotation of an AVL tree with example.	2
10. Explain R-0 rotation of an AVL Tree with example.	2
11. Compare AVL Tree and a binary search tree.	3
12. Discuss the drawbacks of AVL trees.	2
13. Define B-tree with example.	1
14. Discuss the different operations on B-Trees.	1
15. List the properties of B-Trees.	1
16. What are the advantages of B-trees?	1
17. Define B+ tree.	1
18. Explain the procedure to insert a node into B-Tree.	2
19. Compare B-tree and B+ tree.	3
20. State the properties of red black tree.	1
21. Define M-way tree.	1
Long Answer Questions	
1. Describe the insertion, deletion, searching operations on binary search trees.	2
2. Explain the insertion operation on AVL trees.	2
3. State the properties of Red-Black trees with example.	2
4. Explain the searching and deletion operation on AVL Tress.	2
5. Define binary search tree. Construct the binary search Tree for the below given data. P, F, B, H, G, S, R, Y, T, W, Z.	3
6. Explain various rotations of AVL Trees maintaining balance factor while insertion takes	2

place.	
7. Write a C program that uses functions to perform the following:	3
a) Create a binary search tree of integers.	
b) Traverse the above Binary search tree recursively in in-order.	
8. Insert the following elements into an empty AVL Tree20, 15, 5, 10, 12,17,25,19.	3
9. Explain the various rotations of AVL Trees maintaining balance factor while deletion takes place.	2
10. Construct AVL Tree with the following elements C,O,M,P,U,T,I,N,G and remove the elements P , U and T.	3
11. Explain the following with example.	2
a. Red-black trees.	
b. M-Way trees.	
c. B-Trees.	
d. B+ trees.	
12. Define B-tree, B+ tree? What are the advantages and disadvantages of B-tree? Explain with an example.	3
UNIT-4	
Short Answer Questions	
1. Define graph.	1
2. Discuss the representation of graph with example.	2
3. Explain the Adjacency matrix representation of graph.	2
4. Explain path matrix representation of graph.	2
5. Explain the linked representation of the graph	2
6. List the different graph traversals.	1
7. Differentiate BFS and DFS.	2
8. Define Spanning tree.	1
9. Define minimum Spanning tree.	1
10. Explain the basic properties of a spanning tree.	2
11. Define weighted graph.	1
12. Define sub graph.	1
13. Define cyclic graph.	1

14. Define di-graph.	1
15. Define in degree, out degree for a graph.	1
16. Define path in a graph.	1
17. Explain the advantage of DIJKSTRA algorithm.	2
18. List the applications of Graphs.	1
19. Differentiate tree and a graph.	2
20. What is the difference between BFS and DIJKSTRA algorithm?	4
Long Answer questions.	
1. Explain graph ADT.	2
2. Explain different ways representation of graphs.	2
3. Explain BFS graphs traversal algorithms with suitable example.	2
4. Explain DFS graphs traversal algorithms with suitable example.	2
5. Differentiate BFS and DFS.	3
6. Define Graph and explain how graphs can be represented in adjacency matrix and adjacency LIST.	1
7. Write the advantages of using BFS over DFS or using DFS over BFS? What are the applications and downsides of each?	1
8. illustrate BFS and DFS traversals of following graph 	3
9. Illustrate DFS and BFS traversals of following graph 	3
10. Define Minimum Spanning tree? Explain Kruskal's Algorithm with example.	3

11. Define Minimum Spanning tree? Explain Prims Algorithm with example.	3
12. Explain DIJKSTRA Algorithm with example.	3
13. Explain DIJKSTRA Algorithm for the following graph. 	3

UNIT-5

Short Answer Questions	
1. Differentiate linear search and binary search.	2
2. Define Hashing.	1
3. Compare the time complexities of binary search, linear search, hashing.	3
4. Define hash table.	1
5. Define Hash Function.	1
6. List different types of popular hash functions.	1
7. Explain mid square method with example.	2
8. Define Collision.	1
9. Define probe.	1
10. State different types of collision resolving techniques.	1
11. Define Separate Chaining	1
12. Define Open Addressing	1
13. Define Linear probing	1
14. Define Quadratic Probing	1
15. Define Double Hashing	1
16. Define rehashing	1
17. What is the use of extensible hashing	1

18. List the uses of hash table	1
19. Define dictionaries.	1
20. What are the uses of dictionaries?	1
Long answer questions	
1. Define hashing and discuss the different hashing functions with an example.	2
2. Define collision and discuss any two collision resolution techniques	2
3. Explain Chaining with an example	2
4. Explain the implementation of dictionaries using hashing.	2
5. Use quadratic probing to fill the Hash table of size 11. Data elements are 23,0,52,61,78,33,100,8,90,10,14.	3
6. Analyze input (371, 323, 173, 199, 344, 679, 989) and hash function h(x)=x mod 10, Show the result Separate Chaining, linear probing.	4
7. Analyze input (371, 323, 173, 199, 344, 679, 989) and hash function h(x)=x mod 10, Show the result using quadratic probing, and double hashing h2(x)=7 - (x mod 7)	4

Mid Question Papers

Vidya Jyothi Institute of Technology (Autonomous)

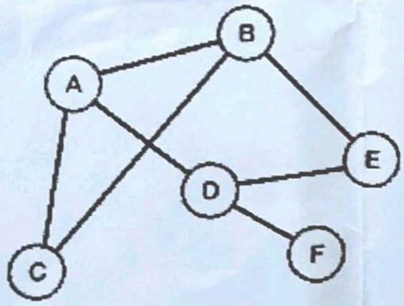
(Accredited by NAAC & NBA, Approved By A.I.C.T.E., New Delhi, Permanently Affiliated to JNTU, Hyderabad)
(Aziz Nagar, C.B.Post. Hyderabad -500075)

II Year B. Tech I Semester Mid-II Examination, February -2022

Subject: Data Structures | **Branch: IT**
Time: 90 Minutes | **Max. Marks: 20**

Bloom's Level:

Remember	L1	Apply	L3	Evaluate	L5
Understand	L2	Analyze	L4	Create	L6

Q.No.	PART-A	BL	CO	PO	Marks
ANSWER ALL THE QUESTIONS	**(3Q x 2M = 6M)**				
1	Define binary search tree with example.	2	3	1,2	2M
2	Discuss the representation of graph with example.	3	4	1,2,3,9	2M
3	Define Hash Function. List various hash functions.	2	5	1,2,9	2M
	PART-B				
ANSWER ALL THE QUESTIONS	**(5+5+4=14M)**				
4. i	Construct AVL Tree with the following elements C,O,M,P,U,T,I,N,G and remove the elements P , U and T.	6	3	1,2,4,9	5M
	(OR)				
ii	Describe the insertion, deletion, searching operations on binary search trees.	4	3	1,2,4,9	
5 . i	Explain DFS graphs traversal algorithms and Illustrate DFS traversals of following graph. 	6	4	1,2,3,4,9	5M
	(OR)				
ii a)	Define Minimum Spanning tree? Explain Kruskal's Algorithm with example.	4	4	1,2,3,4,9	2M
b)	Explain DIJKSTRA Algorithm with example.	4	4	1,2,3,4,9	3M
6. i	Analyze input (371, 323, 173, 199, 344, 679, 989) and hash function h(x)=x mod 10, Show the result using Separate Chaining.	4	5	1,2,4,9	4M
	(OR)				
ii	Analyze input (371, 323, 173, 199, 344, 679, 989) and hash function h(x)=x mod 10, Show the result using linear probing.	4	5	1,2,4,9	

VJIT(A)

Data Structures

Scheme Of Evaluation

II B Tech I Semester Mid-II February 2022

Branch: IT

Subject: DS

Max Marks: 20M

Instructor: K. Shireesha.

Part - A

1. Define binary search with example. 2M.

   Definition - 1M

   Diagram of BST - 1M

2. Discuss the representation of graph with example. 2M

   Graph diagram - 1M

   Different representations - 1M

3. Define hash function. List various hash functions. 2M

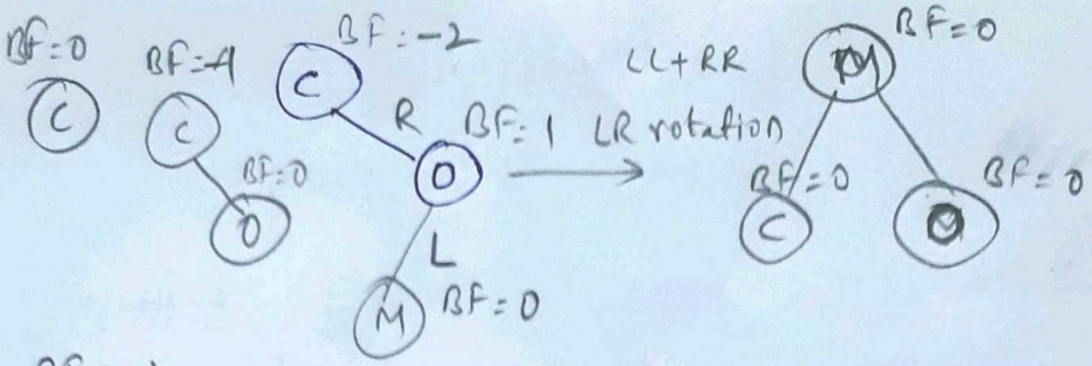
   Definition - 1M

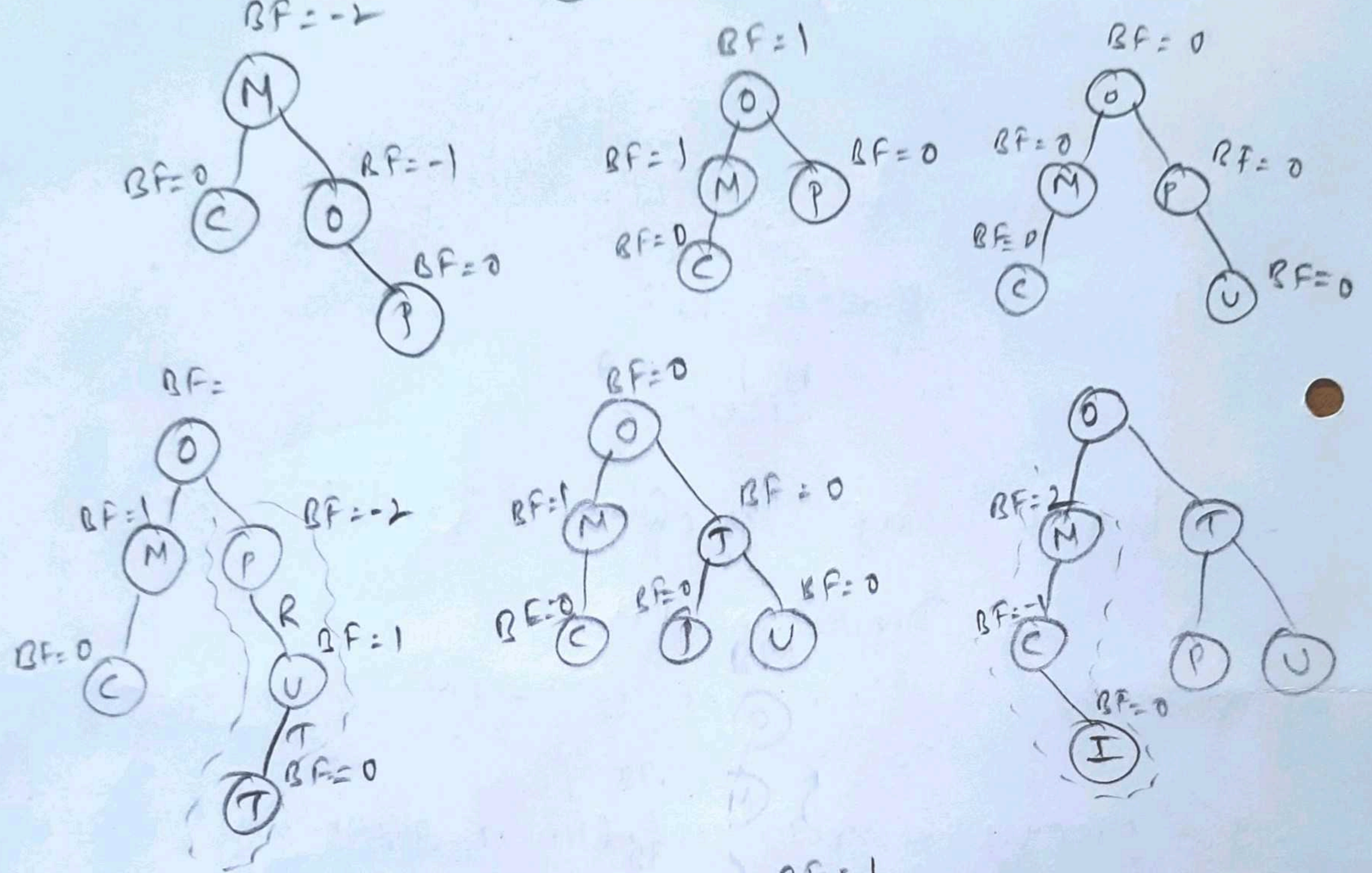
   Different types of hash functions - 1M.

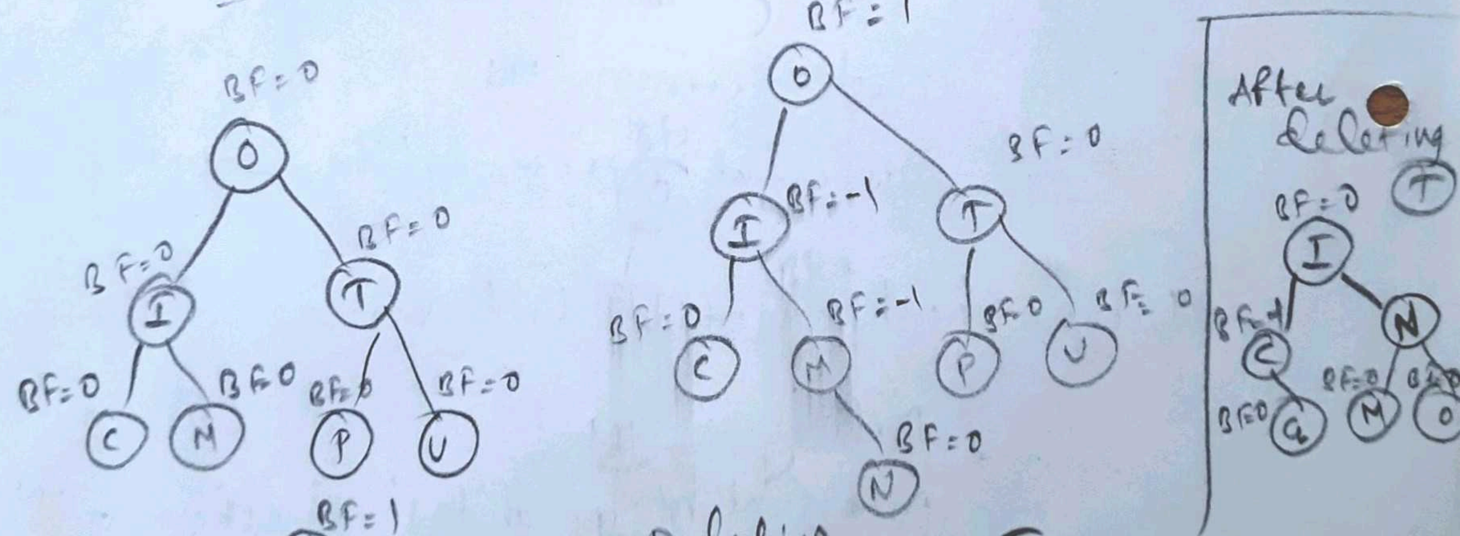
Part - B

4(i) Construct AVL Tree with the following elements C, O, M, P, U, T, I, N, G and remove elements P, U, T — 5M

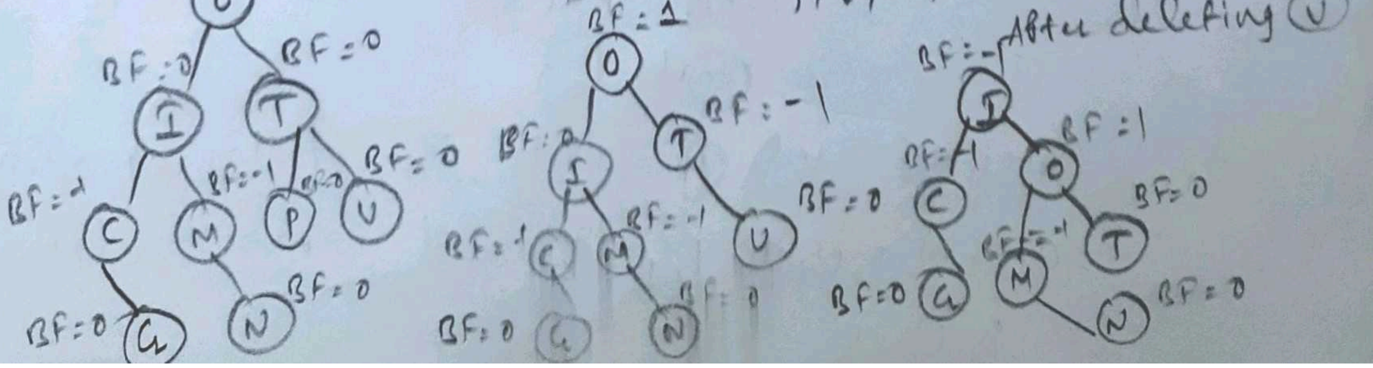
Sol^n:







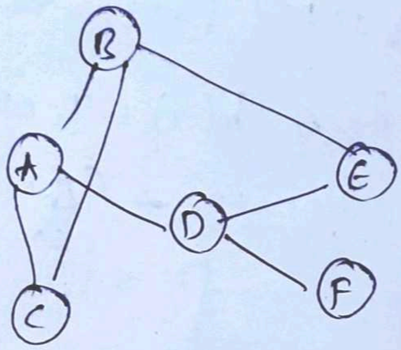
Deletion P, U, T



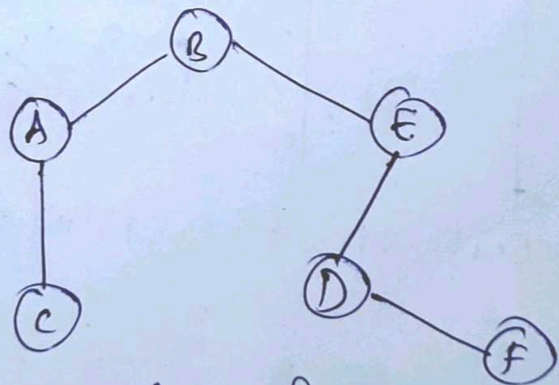
(ii) Describe the insertion, deletion, Searching Operations on binary Search Trees. (5M)

By taking some example need to show insertion – 2M
deletion – 2M
Search – 1M

(5) (i) DFS Graph Traversal algorithms & illustrate DFS traversals of following graph. – (5M)



Writing DFS algorithm – (2M)



construction of DFS for above graph
Step by step procedure – (3M)

(ii) a. Define minimum Spanning tree? Explain Kruskal's Algorithm with example. (2M)

Definition - 1(M)

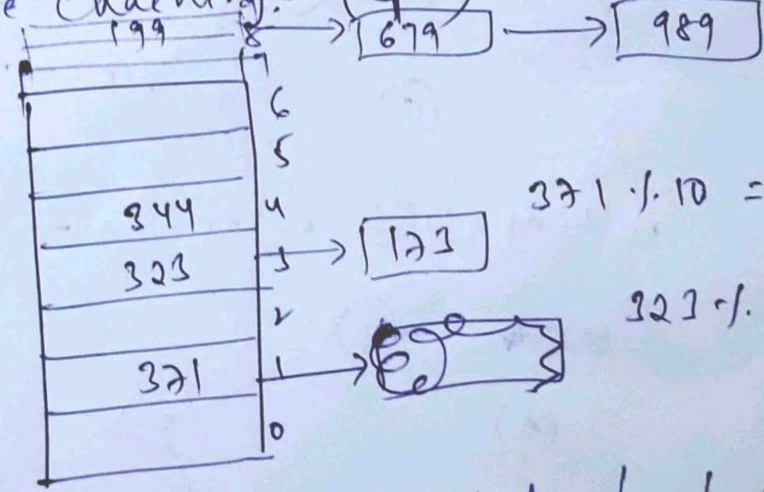
Kruskal's Algorithm - (1M)

b) Explain DIJKSTRA algorithm with example - (3M)

Algorithm writing - 1M

example - (2M)

6 (i) Analyse input (371, 323, 173, 199, 344, 679, 989) and hash function $h(x) = x \bmod 10$. Show the result using Separate chaining - (4M)



371 % 10 = 1

323 % 10 = 3

process of storing into hash table - (4M)

Vidya Jyothi Institute of Technology (Autonomous)

(Accredited by NAAC & NBA, Approved By A.I.C.T.E., New Delhi, Permanently Affiliated to JNTU, Hyderabad)
(Aziz Nagar, C.B.Post, Hyderabad -500075)

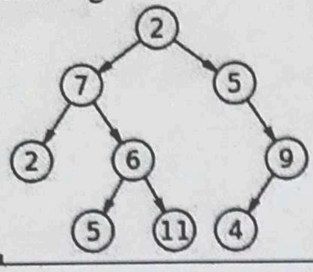
II B.Tech I Semester Mid-I Examination, November-2021

Subject: Data Structures **Branch: IT**
Time: 90 Minutes **Max Marks:20**

Bloom's Level:

Remember	L1
Understand	L2
Apply	L3
Analyze	L4
Evaluate	L5
Create	L6

Q.No.	PART-A	BL	CO	PO	Marks
ANSWER ALL THE QUESTIONS			**(2Q x 3M = 6M)**		
1	Why queue is called FIFO list and stack is called LIFO list.	1	1	1,2,3, 4, 5 , 11,12	3M
2	Define a binary tree. List the types of binary trees	1	2	1,2,3, 4, 5 , 11,12	3M
	PART-B				
ANSWER ALL THE QUESTIONS			**(2Q x 7M=14M)**		
3. i a)	Discuss the algorithms for push and pop operations on a stack.	6	1	1,2,3, 4, 5 , 11,12	3M
b)	Write a program for converting infix expression to postfix expression and explain with an example.	3	1	1,2,3, 4, 5 , 11,12	4M
	(OR)				
ii. a)	Define data structure. Explain different types of data structures	3	1	1,2,3, 4, 5 , 11,12	3M
b)	Write a program to implement Queue ADT	3	1	1,2,3, 4, 5 , 11,12	4M
4 i. a)	Explain binary tree ADT.	5	2	1,2,3, 4, 5 , 11,12	3M
b)	Discuss representation of binary tree using arrays and linked list.	6	2	1,2,3, 4, 5 , 11,12	4M
	(OR)				

ii .a)	What is a threaded binary tree? Explain with an example	4	2	1,2,3, 4, 5 , 11,12	3M
b)	Write in order, preorder, post order traversal of the following tree. 	4	2	1,2,3, 4, 5 , 11,12	4M

VJIT(A)

End Exam Papers

Vidya Jyothi Institute of Technology (Autonomous)

(Accredited by NAAC & NBA, Approved By A.I.C.T.E., New Delhi, Permanently Affiliated to JNTU, Hyderabad)
(Aziz Nagar, C.B.Post, Hyderabad -500075)

R20

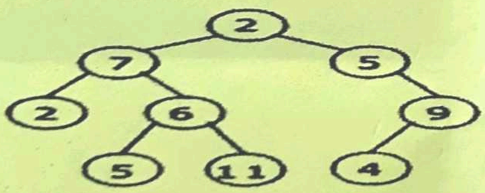
Subject Code:A43503

B.Tech. II Year I Semester Regular Examination, February - 2022

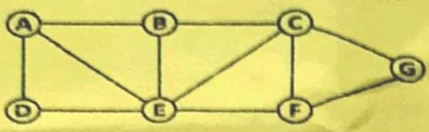
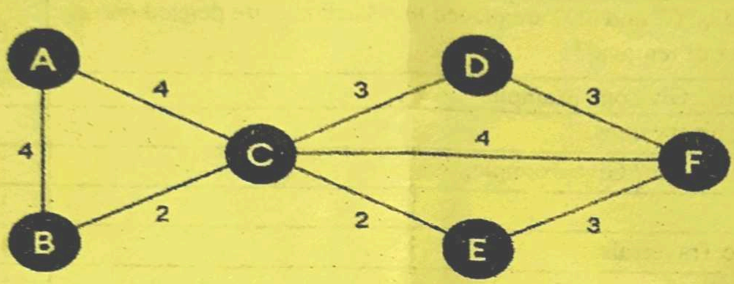
Subject: Data Structures **Branch: CSE, IT & CSE(DS)**
Time: 3 Hours **Max. Marks:75**

Bloom's Level:

Remember	L1
Understand	L2
Apply	L3
Analyze	L4
Evaluate	L5
Create	L6

PART - A ANSWER ALL THE QUESTIONS 25 Marks		CO	PO	Bloom's Level	Marks
1	What is the value of the postfix expression 2 3 6 * 2 / +	1	1-12	L1	2M
2	If the elements "A", "B", "C" and "D" are placed in a stack and are deleted one at a time, what is the order of removal?	1	1-12	L2	3M
3	What is a full binary tree? Give one example.	2	1-12	L1	2M
4	Write the Properties of binary trees.	2	1-12	L3	3M
5	What is a complete binary tree? Give example.	3	1-12	L3	2M
6	What is AVL tree?	3	1-12	L2	3M
7	List and define the tree Traversals.	4	1-12	L2	2M
8	Write about Dijkstra Algorithm.	4	1-12	L3	3M
9	Give the collision resolution techniques.	5	1-12	L6	2M
10	If a hash table of size 11, where is element 7 is placaed. Use the hash function h(key)= key mod tablesize	5	1-12	L2	3M
PART - B					
ANSWER ALL THE QUESTIONS	**5Q X 10M = 50 M**				
11.i	Pictorially illustrate a Circular Queue having size=5 with initial index of front =rear=-1. a) Cir_enqueue (33), b) Cir_enqueue (24), c) Cir_enqueue (15), d) Cir_dequeue (), e) Cir_enqueue (42), f) Cir_dequeue (), g) Cir_enqueue (78), h)Cir_dequeue (), i) .Cir_enqueue (87), j) Cir_enqueue (51), k) Cir_enqueue (6), l) Cir_dequeue () at every step highlight the index of front and rear.	1	1-12	L2	10M
	[OR]				
ii.	Convert the given Infix to post fix expression: ((A - (8 + B)) * 4) ^ (C + D)	1	1-12	L6	10M
12.i	Traverse the following binary tree in pre, post, and inorder.  	1	1-12	L3	10M
	[OR]				
ii.a)	Explain Binary tree representations with an example.	2	1-12	L6	5M
b)	Construct a binary tree from a given preorder and inorder sequence: Preorder: A B D G C E H I F Inorder: D G B A H E I C F	2	1-12	L3	5M

P.T.O

		CO	PO	Bloom's Level	Marks
13.i	Construct a B- tree of order m=3 as maximum height using the sequence of nodes: 50,11,65,26,62,10,55,22,70,35,80,15,85,40	3	1-12	L6	10M
	[OR]				
ii.	Write the properties of Red-Black Trees.Give one example.	3	1-12	L2	10M
14.i.	Create a spanning tree, using BFS traversal of given graph. 	4	1-12	L6	10M
	[OR]				
ii	Find the minimum cost of the following Minimum Spanning Tree using Prim's algorithm. 	4	1-12	L4	10M
15.i	Consider a hash table of size 10. Using quadratic probing, insert the keys 72, 27, 36, 24, 63, 81, and 101 into the table. Use h(k) = (K+K+K) mod m.	5	1-12	L1	10M
	[OR]				
ii.	Insert the following elements into hash table initially empty of size 10 45, 70, 76, 85, 89, 69 and 125 .Use linear probing, quadratic probing, double hashing(R=7) and separate chaining technique for collision resolution.	5	1-12	L6	10M

VJIT(A)

Vidya Jyothi Institute of Technology (Autonomous)

R18

(Accredited by NAAC & NBA, Approved By A.I.C.T.E., New Delhi, Permanently Affiliated to JNTU, Hyderabad)
(Aziz Nagar, C.B.Post, Hyderabad -500075)

Subject Code:A23504

B.Tech. II Year I Semester Examination NOVEMBER -2019

BRANCH : CSE&IT

SUBJECT : DATA STRUCTURES

Max. Marks:75

Time: 3 Hours

Note: This question paper contains two *Parts A and B.*

Part A is compulsory which carries 25 Marks. Answer all question in Part A.

Part B consists of 5 questions. Answer all the questions.

Bloom's Level:

Remember	L1	Analyze	L4
Understand	L2	Evaluate	L5
Apply	L3	Create	L6

	PART - A	Bloom`s Level	25 Marks
	ANSWER ALL THE QUESTIONS	L1	2M
1	What is Data Structure? Explain its types.	L4	3M
2	Distinguish statcks and queues.	L1	2M
3	Define full binary tree.	L1	3M
4	Define path,sibling,height of tree with an example	L4	2M
5	Explain the necessity of height balancing in trees.	L4	3M
6	Explain single rotations in insertion of AVL Tree.	L4	2M
7	List any two differences between graphs and trees.	L5	3M
8	Briefly explain DFS Graph Traversal.	L2	2M
9	Differentiate Linear Probing and Quadratic Probing.	L5	3M
10	Explain Dictionaries with an example		

	PART - B	Bloom`s Level	50Marks
	ANSWER ALL THE QUESTIONS		
11.i.a)	Convert the infix expression a / b – c + d * e – a * c into postfix expression and trace that postfix expression for given data a=6,b=3,c=1,d=2,e=4.	L6	6M
b)	Discuss an algorithm to insert an element in a queue.	L6	4M
	[OR]		
ii.a)	Explain advantages of circular queue over linear queue and explain its operations.	L2	6M
b)	Convert the following expression into potfix notaion. A + B * C + D – E + F	L6	4M
12.i.a)	Explain the sequential representation of a binary tree.	L2	5M
b)	Define extended binary tree. Explain tree traversal procedure of extended binary tree.	L1	5M
	[OR]		
ii.a)	What operations can be performed on binary trees? Discuss.	L1	5M
b)	Explain in-order traversal of a binary tree. Explain with example.	L4	5M
13.i.a)	Construct an AVL tree and update the height and balance factor after every insertion for the following elements 14, 17, 11, 7, 53, 4, 13, 12, 8 and remove the elements 53, 11 and 8.	L3	6M
b)	Explain Red-Black trees with appropriate example.	L4	4M
	[OR]		
ii.a)	Write a program to illustrate the insertion of a node in the Binary Search Tree	L3	5M
b)	Explain B-tree with example.	L4	5M
14.i.a)	What is graph? Explain the properties of graphs.	L1	5M
b)	Write Breadth First search traversal algorithm. Explain with an example.	L3	5M
	[OR]		
ii.a)	Write Kruskals Algorithm.	L3	5M
b)	Write Dijkstra Algorithm.	L3	5M
15.i.a)	What is hashing? Explain Double Hashing and Rehashing.	L1	5M
b)	Explain Separate Chaining and Open Addressing.	L4	5M
	[OR]		
ii.a)	Write short notes on hashing functions.	L3	5M
b)	Explain about extensible hashing.	L4	5M

VJIT(A)

Content Beyond Syllabus

Content beyond Syllabus

S. No	Name of the Topic
1	Priority Queues and Sorting Techniques
2	Pattern Matching Algorithms
3	Time Complexities of various Algorithms

Unit Wise PPTs and Lecture Notes

'19 June 2019'

Data Structures

Defination: It is a Logical way of storing the data and it derives how the data can be retrived.

Data type: Data type is a way to classify the type of the data. There are 2 types of data types.

i) Built in datatypes / primary / primitive.

ii) Derived data types

i) **Built in data types:**

Built in data types are already defined by the Programming Language (or) for which a Language has built in support.

Eg: In 'c' → int, float, char

ii) **Derived data types:** These are built in data types having associated operation.

Eg: Array, Structure, Union, Stack, Queue.

Derived data types are also called as abstract data types (ADT)

Abstract data type (ADT):

It is a derived data type defined by built-in data type with associated operations.

Eg: int a[10]

insert

delete

display

Sort

Search

Abstract data type is also known as Data Structure

We have two types of data Structures.

i, Linear

ii, Non-Linear

i, Linear: In Linear data structures, the arrangement of the data is in Linear format i.e, the storing of the data will be done one after the another

Eg: Array, stack, Queue, Circular Queue Etc, List

ii Non-Linear: The arrangement of the data Elements is in random fashion.

Eg. Tress, graphs, Heap Etc

Based on Storage:

i, Static Data Structures:- Static Data structure is given a fixed area of memory.

It is not possible to Expand the size at runtime, So that Location of Each Element is fixed.

We represent the Elements with array.

Advantages:-

i Faster accessing of the Elements.

ii The memory Locations are created at the time of declaration.

Disadvantages:-

i Non effective use of resource i.e the memory is wasted when we are not using the allocated value

ii Size is not flexible.

iii Dynamic Data Structures:-

In dynamic data structures, the size of the allocation area is flexible i.e, it is possible to Expand the allocation at run time. We represent these Elements with List.

Advantages:

i Effective ~~array~~ usage of resources

ii Flexible size

Disadvantages

i slower access.

'21 June 2019'

Linear data structure:

(1) STACK:

A stack is a Linear data structure where the insertion and deletion can be done at one End.

Stack word based on the principle of LIFO

* It is an abstract data type

Examples

(1) A stack of cups

(2) A stack of Cateteria trays

(3) A stack of Coins.

Operations in stack:

→ Stack can be implemented in 2 ways.

(1) with the help of array (static Representation)

(2) with the help of List (dynamic representation)

Operation

1) push → Used to insert the data Element

2) pop → delete the data Element.

3) Traverse → moving from one point to another poi

4) is full: To know the stack is full or not

5) is Empty: To know " " is Empty.

6) peek: : Used to return the Top most Element.

Over flow:

If the stack is full still we are inserting Element in Such case the data will be over flow.

Under flow:

If the stack is Empty still we are deleting Element in Such Case the data will be under flow.

* Implementation of Stack:

PUSH:

⇒ Algorithm for inserting an Element into the stack.

Algorithm push ()

i) if top > = size then

1) print "stack" is full

ii) Else

1) Read item

2) Top = Top +1

3) Stack [Top] = item

iii) End if

iv) Stop

(6)

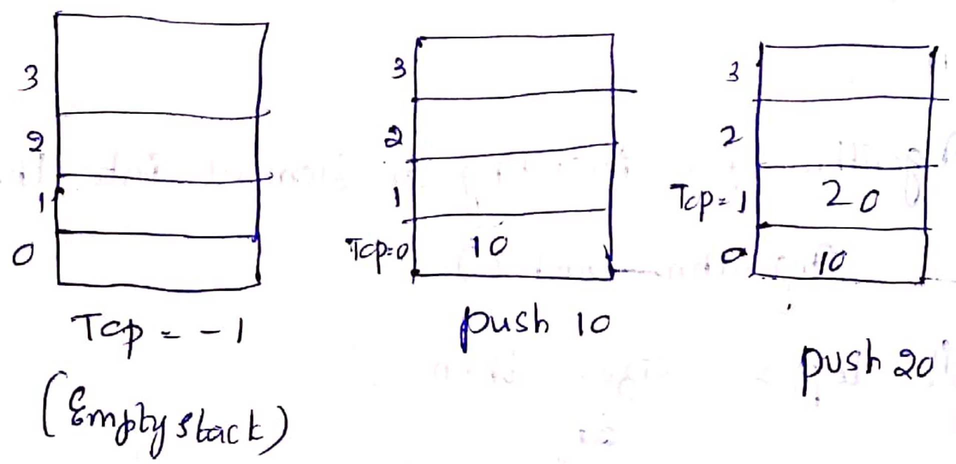
* The first step of this algorithm check for an over flo[w] Condition.

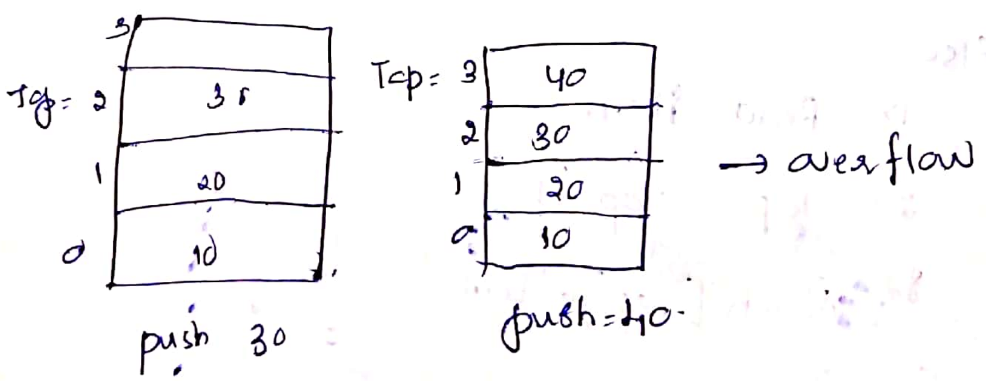
* Over flow means inserting an item into a stack which is full.

* It is the Top Value Reaches to maximum Size of [the] Stack, then items cannot be inserted into the Stack i.e, stack is full.

* Other wise top is incremented by one and item is inserted into the Stack.

Ex Size is 4





Pop: (Delete)

Algorithm for deleting Element from stack.

→ Algorithm pop ()

i) if top = -1 then

    print "Stack is Empty"

2) Else

    i) item = Stack [Top]

    ii) top = Top - 1

    iii) End if

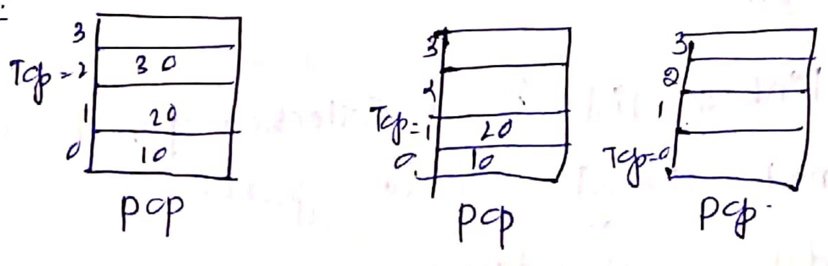
    iv) Stop

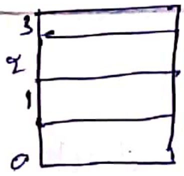
* The first step of this algorithm checks for Underflow Condition.

* If the top value is -1 then Stack is known as Underflow (or) Empty.

* Take out the Element from the Location where the top is pointing and Store it in the Variable then decrement top by one.

Ex.:





pop Top= -1 Underflow.

Display (Traversing)

=> Algorithm display ()

1) if Top= -1

print 'Stack Empty'

2) Else

for (i= top; i> =0; i--)

print Stack [i]

3) Endif

4) Stop

if the Top value is -1 then the stack is Empty. If Stack is not Empty, assign top value to variable i, display the item which is pointed at Stack [i] and decrement the Top value by one and Repeat this until top becomes zero.

22 June 2019

Implementation of Stack Using Dynamic Represe?

Linked List

A Linked list is a collection of nodes and a node consists of two parts.

1) data 2) address.

These are 2 types of Linked list.

1) Single Linked List.

2) Double linked list.

Single LL:

```
Struct node
{
int a;
struct node * next;
};
```

→ A pointer which References to the Same structure then those structures are called as "Self-Refrential structure."

Creation of new node:

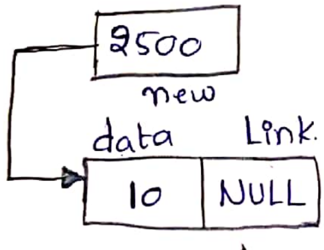
```
Struct node
{
   int a;
   Struct node * next;
};

Struct node *new;

(Struct node *) malloc ( Size of struct (node))
  New → a = 10
  New → next = NULL;
```

10

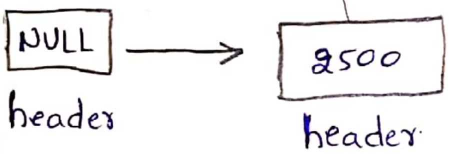
24 June 2019

Inserting a node at begining

Create a node that is to be inserted.



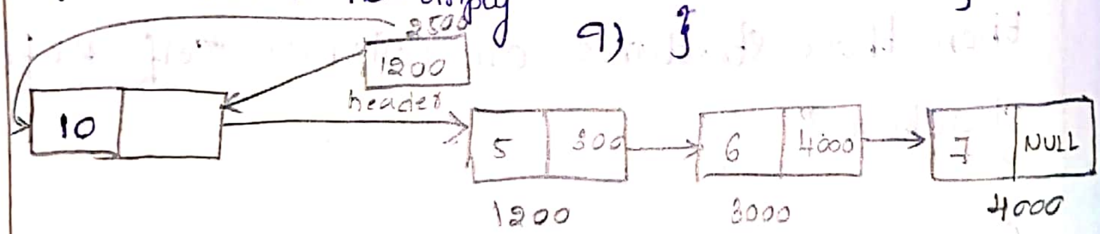
If the list is Empty



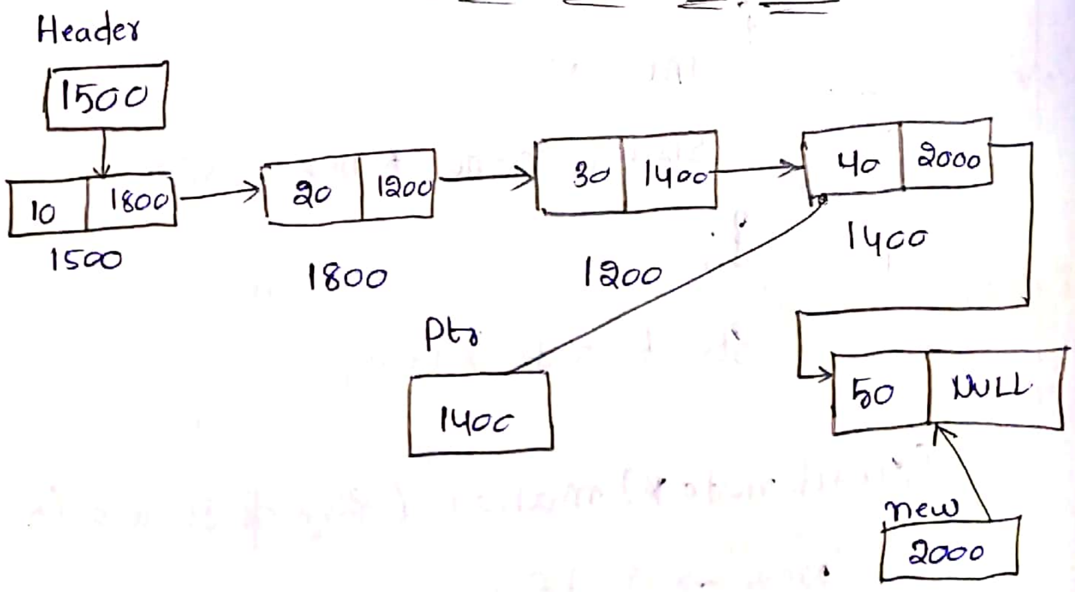
Algorithm

1) Create a new node
2) If (header == NULL)
3) new → Link = NULL;
4) header = new;
5) else
6) {
7) new → Link = header;
8) header = new;
9) }

If the List is not Empty



Inserting the node at End.



Algorithm:

1) new = malloc (size of struct node));

2) ptr = header;

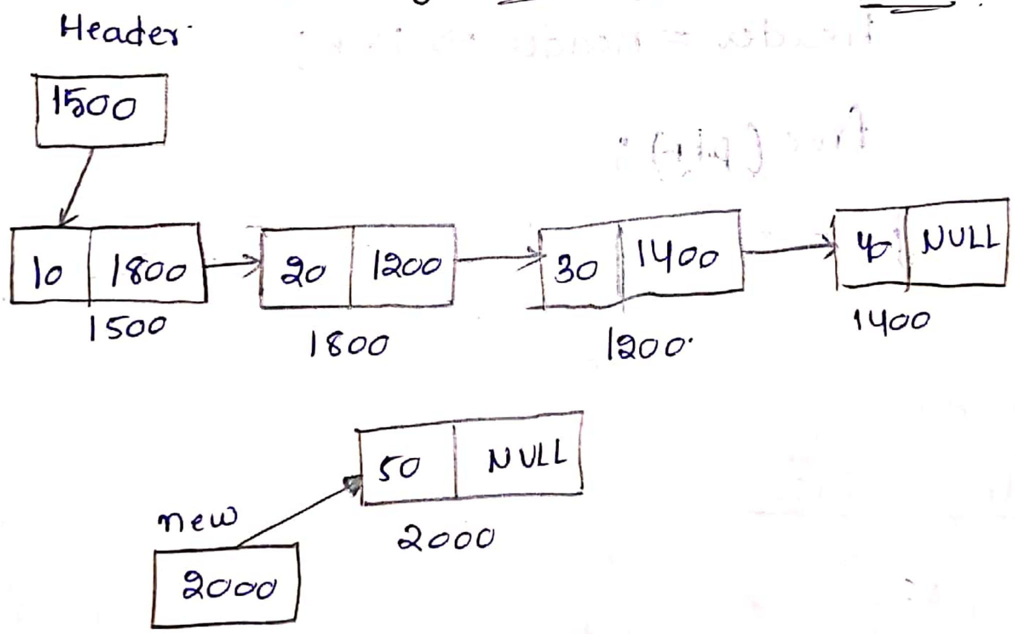
3) while (Ptr → Link! = NULL)

4) ptr = ptr → link

5) new → Link = NULL;

6. ptr → Link = new.

Inserting at the middle: (a) position.



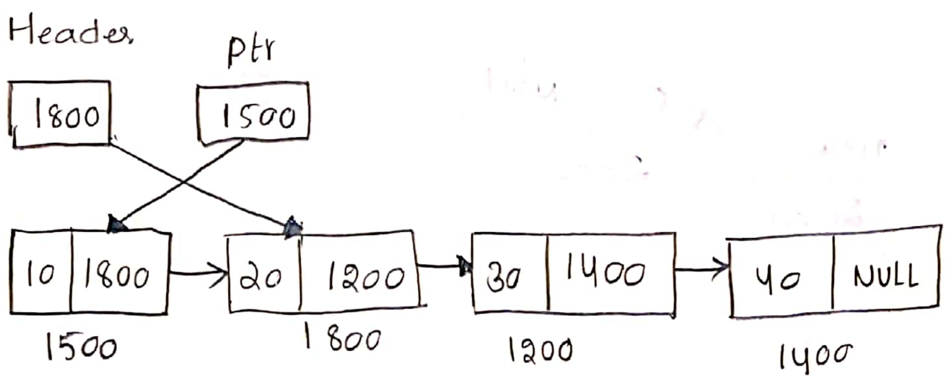
26/06/19

Deletiong a node at the begining

Algorithm:

1) if (header == NULL)

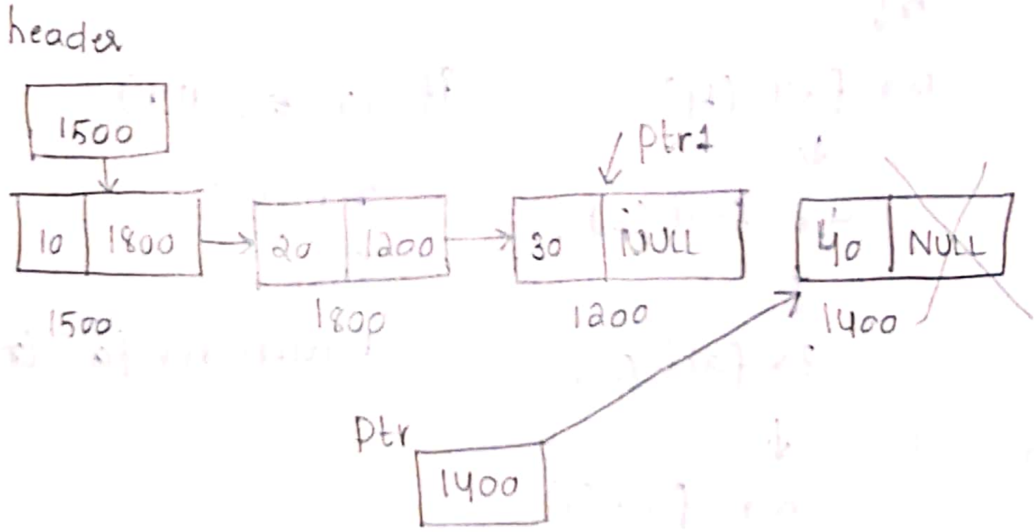
```
{
  printf(" List is Empty");
}
Else
{
  ptr = header;
  header = header -> link;
  free (ptr);
}
```



Deleting the node from the End

Algorithm:

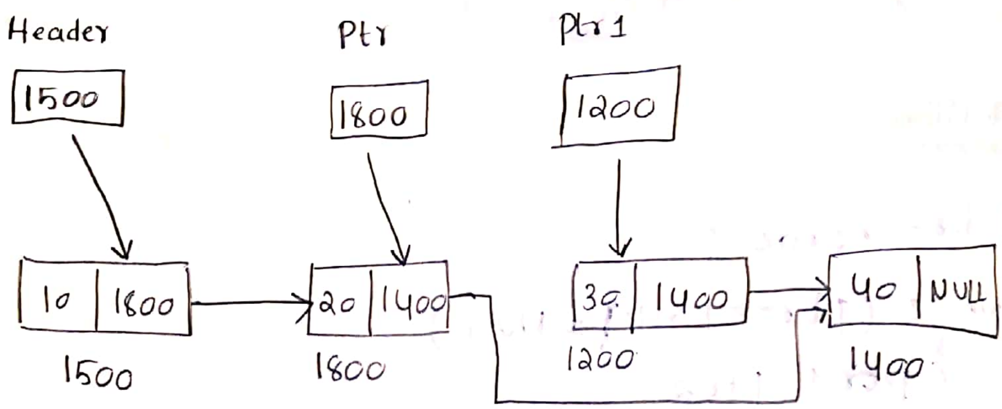
1) ptr = header;
2) while (ptr -> Link != NULL)
3) { ptr1 = ptr2
4) ptr = ptr -> Link; }
5) ptr1 -> Link = NULL;
6) free (ptr);



Deleting a node at given position:

Algorithm:

1) ptr = header
2) for (i=1; i< pos - 1; i++)
3) ptr = ptr -> Link;
4) ptr1 = ptr -> Link;
5) ptr -> Link = ptr1 -> Link;
6) free (ptr1);



Applications of Stack:

1) Recursion

Eg: 5!

5 * fact (4).

↓

4 * fact (3).

↓

3 * fact (2)

↓

2 * fact (1)

1

1

if (n== 0 || 1)

return 1

Else

return n * fact (n-1)

27/06/19

In Notation, An Arithemetic Expression can be written in 3 different notations. These notations are.

1> Infix notation
2> prefix notation
3> Post fix notation

Left to right { () - high
/* % - medium
+, - - Low }

Infix notation:

In this notation operands are having operators those are in between operands.

Ex:- a+b-c

a, b, c are operands

+, - are operators

pre-fix Notation: (Polish Notation)

In this Notation operator is prefix to operands i.e operator is written before operands.

Eg: a+b → +ab

a+b*c → +a*bc

It is also Called as polish Notation.

6*7 → *67

a+b/c → +a/bc

a*b+c → *ab+c
+*abc

$a+b-c \rightarrow +ab-c$

$-abc$

$a+b*c-d \rightarrow a+*(bc)-d$

$+a*bc-d$

$-+a*bcd$

$\rightarrow a/b*c-d. \rightarrow (A*B)+(C*d)$

$-*/abcd. \quad +*AB*Cd$

$\rightarrow ((a+b))*c)-d$

$-*+abcd.$

Postfixnotation:

If this notation this operator is post fixed i.e, the returned after the operands this is also known as Reverse polish notation

Ex: $a+b \rightarrow a+b*c$

$ab+ \rightarrow abc*+$

Convert given Expression into postfix.

1) $a+b-c*d$

$a+b-cd*$

$ab+cd*-$

2) $a/b-c+d$

$ab/c-+d$

$ab/c-d+$

28/06/19

Convert the given Expressions to postfix and pre-fix Notation.

~~Arithemetic~~ Expressions can be Expressed in three ways)

((A+B) * (C-d))/e.

((AB + *(c~~d~~d))/e}. /* + AB -cde.

Examples

Infix Expression	Pre-fix Expression	post fix Expression
2+3	+23	23+
2+3*5	+ 2*35	235 *+
(2+3) * 5	*+ 235	2 3+5 *
(2*3) -(5+9)	- * 23+59	23 * 59 +-

Conversion of a given Infix Expression to post fix Expression.

Rules:

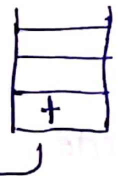
1) The Expression will be read from Left to right. An Empty stack is taken to add operators.

Ex: a+b

P+ String

ab

↳ ab

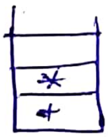


Top = -1

Ex: a+b*c

String

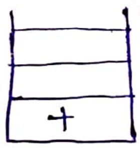
abc * +



Ex: a*b+c.

pf

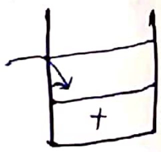
a~~b~~ b*c+



If the Reading symbol is operator then push it into stack. However first pop the operators which already on the stack that have higher or Equal pre than the current operator

Write an Expression a-b*c/d+e%f

abc*d/-ef%+

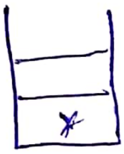


If the Reading Symbol is open or Left paranthesis then, push into Stack.

If the Reading Symbol is Right or Closed paranthesis then pop all the Contents of stack until Respective Left paranthesis is poped. And add all the poped Element to post fix String

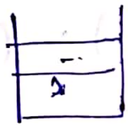
(a+b)*c.

ab* c



(a+b-c)*d.

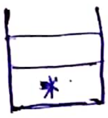
ab+c-d*



Conver the ~~using~~ given Expression into postfix using Stack.

((a+b-c)*d)/e

ab+c-d*e/



7- (2*3 +5)*(8-4/2).

Remaining Infix String	Stack	Postfix
7-(2*3+5)*(8-4/2)	Empty	null
-(2*3+5)*(8-4/2)	Empty	7
(2*3+5)*(8-4/2)	-	7
2*3+5)*(8-4/2)	-(	7
3+5)(8-4/2)	-(	72
3+5)*(8-4/2)	-(*	72
+5)*(8-4/2)	-(*	723
5)*(8-4/2)	-(+	723*
)*(8-4/2)	-(+	723*5
*(8-4/2)	-	723*5+
(8-4/2)	-*	723*5+
8-4/2)	-*(	723*5+
-4/2)	-*(	723*5+8
4/2)	-*(-	723*5+8
/2)	-*(-	723*5+84
2)	-*(-/	723*5+84
)	-*(-/	723*5+842

null Empty 723* 5+842 /-*-

29/06/19

Conversion of Infix to pre-fix Using stack :

1) Reverse the given Infix Expression

Ex (A+B)*C

C*B+A

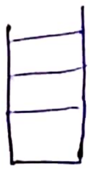
while Reversing Each "(" becomes ")" and ")" becomes "("

2) Obtain the postfix Expression of the Modify Infix Expression

3) Reverse the postfix Expression.

Ex: ① (A+B)*C

C*(B+A)



Postfix:

CB* A+

prefix

+A*BC.

② (A+B)*C.

C*(B+A)

PS

CBA +*

prefix: *+ABC

(22)

Convert the given Expression

* $(((A+B)*C-d)/e)*f$.

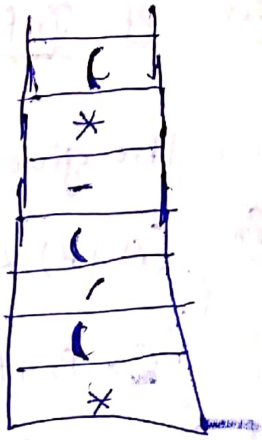
$((+AB*C-d/e)*f$.

$(*+ABC-d/e)*f$

~~post~~ * ~~/*~~+ABC Cdef

post $f*(e/(d-C*(B+A))$

fedCBA+*-/*.



Pref. */-*+ABCdef.

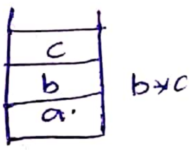
1/07/19

Evaluation of post fix Expression to result:

a+b*c

a+bc*

abc*+



b*c

$\overset{5}{a}+b$ ~~*~~ $\overset{12}{c}$.

17

Evaluation of Algorithm to Reverse Polish Notation

1) Initialize Empty stack

2) Convert the given Infix Expression into R.P.N

3) If a token is an operand push it on the stack.

4) If a token is an operator ~~pop~~ apply the following steps

→ pop the Top two values from the Stack. apply Operator Token in between Top two values in such a way that Top value becomes operand 2 and Top -1 value becomes operand 1.

→ push the Result on to the Stack

5) Repeat Step 3 and 4 until the End of Expression ~~Ste~~

6) The value of the Expression on the Top of the Stack. print the Result as Top value.

Ex: Evaluate the postfix Expression 5, 6, 2, +, *, 12, 4, /, -

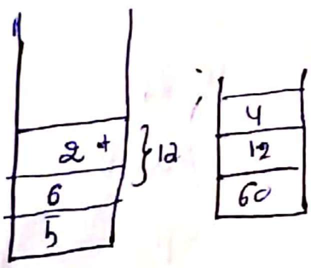
Ans

6+2.

8 ×5.

40 - 3

= 37.



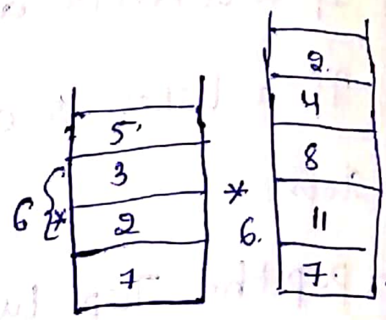
Evaluate the following Exp $7-(2*3+5)*(8-4/2)$

$7-(2*3+5)*(8-4/2)$

~~7•(23~~ $(723*5+)*(842/-)$

$723*5+842/-*-$

$\Rightarrow -59$



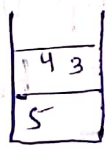
Evaluation of prefix Expression:

Ex $A+B*C$ — $A=3$

$+A*BC$ — $B=4$

$+3*45$ — $C=5$

$=23$



1) Convert the given Infix Expression into prefix Expression.

2) Start Scaning the String from right to left 1 character at a time.

3) If it is an operand push it onto Stack.

4) If it is an operator, pop Top two operates from Stack in such away that top value becomes operand 1 and TOP-1 value becomes operand 1

5) perform the operation and push the result into the sta

⑥ Repeat ③ & ④ upto the End of the string.

⑦ Display The result is stored in the Top and it is displayed.

E.

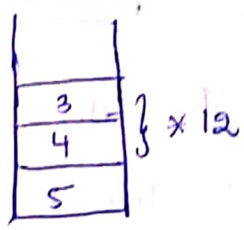
Convert the given Expression into pre-fix.

a+b-c * d/e

a=1 d=4.

b=2 e=5.

c=3



a+b-/* cde

- + ab/* cde.

- +12/* 345

Queue:

Queue is an abstract data type where the insertion and deletion can be done from different Ends.

The Insertion End is called ~~Ray~~ Rear and Deletion End is Called front.

Queue follows the principle of first in first out (FIFO

Basic queue operations:

Insertion: To add a new item to the rear End of the Queue The rear End always points to the Recently added Element.

Deletion: To delete the item from front End of Queue The front always points to the Recently removed Element.

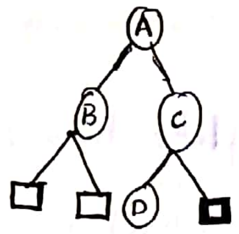
Display: To display items of Queue.

Example:

1) Printing the papers.

Algorithm:

Ex:



Dummy Nodes Represents Square boxes with consists of NULL data.

31/07/19

Threaded binary Tree:

In a linked Representation of a Binary tree a number of nodes Contain a NULL pointer either in their left or Right field or in both. This page is wasted in storing a Null pointer Can be used to store Some other information.

→ The Null pointers Can be Replaced to store a pointer to the Inorder predecessor or The in order Successor of the Node. These Special pointers are called threads and Binary trees Containing threados are Called Threaded binary tree.

There are two ways to Construct threaded binary trees

→ One way threading

→ Two-way threading.

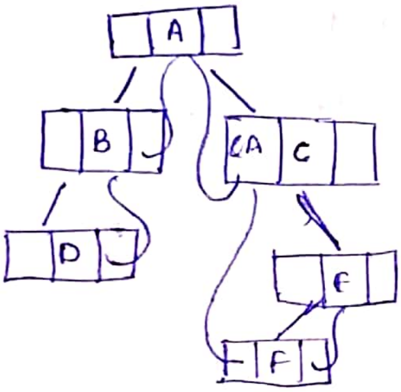
One way Threading:

In a threaded binary tree if the threads appeard in the Left field (In order predecessor). Then that threading is Called Left threading

Two way threading:

In a threaded binary tree if the thread appeard in both the Right field & Left field (Inorder Sucessor). Then that threading is Called ~~Right threading~~. Two way threading.

Ex:



31/07/19

3. Advanced Concepts of trees

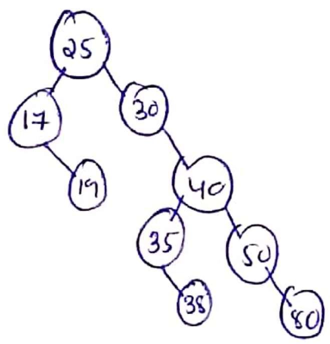
Binary Search tree :

→ A Binary Search tree is a binary tree in which the nodes are arranged in an order.

→ All the nodes in the left Subtree have a Values less than that of the root node. All the nodes in the Right Sub tree have a value greater than the Root node.

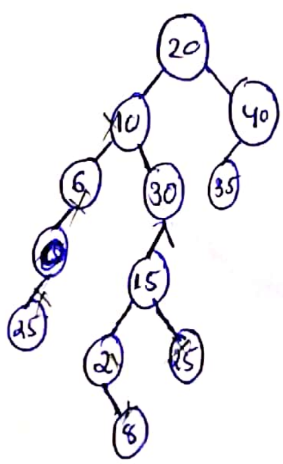
→ If it is Equal It is placed on the Right Sub tree.

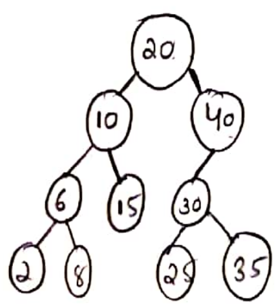
Eg : 25, 30, 40, 50, 80, 17, 19, 35, 38



Construct a Binary Search tree with the following Element

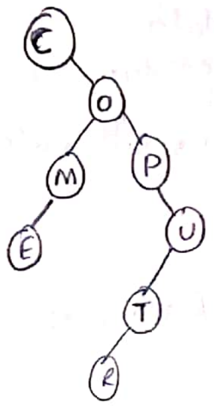
20, 40, 10, 30, 6, 15, 25, 2, 8, 35





Construct a Binary Search tree with following letters

COMPUTER



2/08/19

Basic operations on Binary Search tree:

1) Search ( ) : It finds an Element in a tree.

2) Insert ( ) : It inserts an Element in a tree.

3) Traverse ( ) : Traversing from one node to another tree.

4) delete ( ) : It Removes an Element from the tree.

Node:

Defining a node having Some data, references to it Left and Right child node.

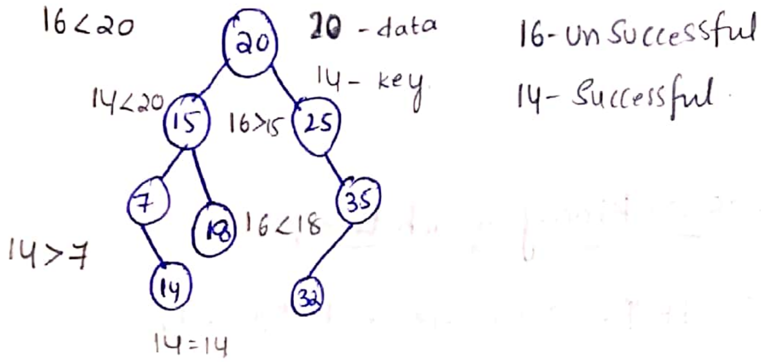
```
Struct node
{
int data ;
Stuct node * left ;
Stuct node * right ;
};
```

① Search Operation:

→ Whenever an Element is to be Searched start Searching from the Root node.

→ If the ~~data~~ keyvalue is Less than the ~~key value~~ data. Search for the Element in the ~~Right~~ left Sub tree, Otherwise Search for the Element in the ~~left~~ Right Sub tree.

→ Follow the Same Algorithm for Each node.



Insert Operation:

→ Whenever an Element is to be inserted first Locate its proper Location. start Searching from the Root node if the key value Less than Root data, then Search for Empty Location on Left Sub Tree and insert the data Otherwise Search for Empty location in the Right Sub tree.

Procedure to Search and Insert a node in Binary Searchtree

→ Starting from the Root,

① check whether the value in root node and Searched Value are Equal if So Value is found.

② If Searched Value is Less than the node Value.

ⓐ If current node as no Left child, searched Value does not Exsist in the binary Search tree ⓑ Otherwise handle the Left child with Same Algorithm.

③ If Search value is Greater than Node value.

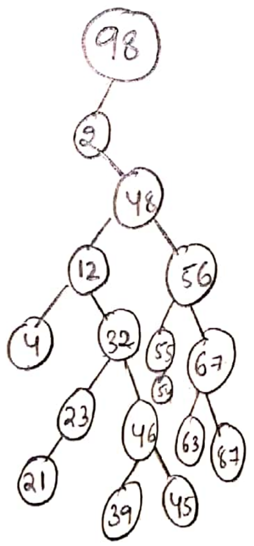
ⓐ If Current node has no Right Child, Searched value does not Exsist in the binary Search tree then insert a node.

ⓑ otherwise handle the Right child with same Algorithm

Create a Binary Search tree with the input given below

98, 2, 48, 12, 56, 32, 4, 67, 23, 87, 55, 46

i) Insert 21, 39, 45, 54, 63



Inorder: 2, 4, 12, 21, 23, 32, 39, 45, ~~48~~ 46, 48, 54, 55, 56, 63, 67, 87, 98

For a Binary Search tree the inorder give ascending Order of the Element.

Title : AVL TREE

Date :
Page No. :

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int key;
    struct node *left;
    struct node *right;
    int height;
};
int max (int a, int b);
int height (struct node *N)
{
    if (N == NULL)
        return 0;
    return N->height;
}
int max (int a, int b)
{
    return (a>b)? a:b;
}
```

Title :

Date :
Page No. :

// helper function that allocates a new node with the given key and NULL left and right Pointers

```
struct node * newNode (int key)
{
    struct node * node = (struct node *) malloc (sizeof (struct node));
    node -> key = key;
    node -> left = NULL;
    node -> right = NULL;
    node -> height = 1;    // new node is initially added at leaf
    return (node);
}
```

// A utility function to right rotate subtree rooted with y

```
struct node *rightRotate (struct node *y)
{
    struct node *x = y ->left;
    struct node *T2 = x -> right;
    // Perform rotation
    x ->right = y;
    y -> left = T2;
```

Title :

Date :
Page No. :

```
// update heights
y -> height = max (height (y->left), height (y->right)) +1;
x -> height = max(height (x->left) ,height(x->right))+1;

// Return new root
return x;
}

// A utility function to left rotate subtree rooted with y

struct node *leftRotate (struct node * x)
{
    struct node *y = x ->right;
    struct node *T2= y->left;

    // perform rotation
    y->left = x ;
    x->right =T2;
    // update heights
    x -> height = max(height (x->left) ,height(x->right))+1;
    y -> height = max (height (y->left), height (y->right)) +1;

    // Return new root
    return y;
}
```

Title :

Date :
Page No. :

```
// get balance factor of node N
int getBalance (struct node *N)
{
    if (N==NULL)
        return 0;
    return height (N-> left) - height (N-> right);
}
struct node* insert (struct node* node, int key)
{
    //1. Perform the normal BST rotation
    if (node = = NULL)
        return (newNode(key));
    if (key<node -> key)
        node -> left = insert (node -> left , key);
    else
        node -> right = insert (node -> right, key);
    // 2. update height of this ancestor node
    node -> height = max (height(node->left), height (node -> right))+1;
    // 3. Get balance factor of this ancestor node to check whether
       this node becomes unbalanced.
    int balance = get Balance (node);
```

```
// If this node becomes unbalanced, then there are 4 cases

// Left Left Case
if (balance >1 && key < node -> left -> key)
    return rightRotate (node);

// Right Right Case
if (balance <-1 && key > node -> right -> key)
    return rightRotate (node);

// Left Right Case
if (balance >1 && key > node -> left -> key)
{
    node -> left = leftRotate (node -> left);
    return rightRotate (node);
}

// Right Left Case
if (balance <-1 && key < node -> right -> key)
{
    node -> right = rightRotate (node -> right);
    return leftRotate (node);
}
// return the (unchanged) node Pointer
return node;
}
```

Title :

Date :

Page No. :

```
Void PreOrder (struct node * root)
{
    if (root != NULL)
    {
        Printf ("%d\n", root->key);
        PreOrder (root->left);
        Preorder (root->right);
    }
}
Void main ()
{
    int data, ch;
    struct node *root = NULL;
    while (1)
    {
        Printf (" 1. Insertion 2. Display 3. Deletion 4. Exit\n");
        Printf ("enter your choice\n");
        scanf ("%d", &ch);
        Switch (ch)
        {
            case 1 : Printf ("enter the Value to be inserted\n");
                     scanf ("%d", &data);
                     root = insert (root, data);
                     break;
```

Title :

Date :
Page No. :

```
case 2 :  Printf("Pre order traversal of the given tree is : \n");
          PreOrder(root);
          break;
case 3 :  exit(0);
  }
 }
}
```

Output:

1. Insertion 2. Display 3. Exit

Enter your choice

1

enter the value to be inserted

10

20

30

40

50

25

Enter your choice

2

Pre order traversal of the given tree is :

30 20 10 25 40 50

Title : Breadth First Search

Date :

Page No. :

```
#include <stdio.h>

int  a[20][20], q[20], visited[20], n, i, j, f=0, r=-1;

int  b[20][20];

void  bfs (int v)

{
    for( i=1;  i<=n; i++)
        if (a[v][i] && !visited[i])
        {
            q[++r]=i;
            visited[i]=1;
            b[v][i]=1;
        }
    if (f<=r)
    {
        bfs (q[f++]);
    }
}
```

Title :

Date :

Page No. :

```
Void main()
{
    int v;
    printf("\n enter the no.of vertices\n");
    scanf("%d", &n);
    for(i=1; i<=n; i++)
    {
        q[i]=0;
        visited[i]=0;
        for(j=1; j<=n; j++)
        b[i][j] =0;
    }
    printf("\n Enter graph data in matrix form:\n");
    for(i=1; i<=n; i++)
    {
        for(j=1; j<=n; j++)
        {
            scanf("%d", &a[i][j]);
        }
    }
```

Title :

Date :
Page No. :

```
Printf("\n Enter the starting vertex: \n");
scanf("%d", &v);
visited[v] = i;
bfs(v);
Printf("\n The nodes which are reachable are:");
for(i=0; i<=n; i++)
{
    if(visited[i])
    Printf("%d \t", i);
    else
    {
        Printf("\n BFS is not Possible. All nodes are not
                reachable");

        break;
    }
}
Printf("\n Spanning Tree matrix: \n");
for( i=1; i<=n; i++)
{
    for(j=1;j<=n;j++)
    Printf("\t %d", b[i][j]);
    Printf("\n");
}
}
```

Output :

Enter the no. of Vertices : 5

Enter graph data in matrix form :

0 1 1 0 0

1 0 0 1 1

1 0 0 1 0

0 1 1 0 1

0 1 0 1 0

Enter the starting Vertex : 1

~~Bfs is not Possible~~

The nodes which are reachable are : 1 2 3

Bfs is not Possible. All nodes are not reachable

Spanning tree matrix :

0 1 1 0 0

0 0 0 1 1

0 0 0 0 0

0 0 0 0 0

0 0 0 0 0

Scanned by CamScanner

Title: Depth First Search

Date :
Page No. :

```
#include <stdio.h>
void DFS(int);
int G[10][10], visited[10], n;
int b[10][10];
void main()
{
    int i,j,r;
    printf("Enter number of vertices :\n");
    scanf("%d", &n);
    printf("\n Enter adjacency matrix of the graph:");
    for(i=1; i<=n; i++)
    {
        for(j=1; j<=n; j++)
        {
            scanf("%d", &G[i][j]);
        }
    }
    for(i=1; i<=n; i++)
        visited[i]=0;
```

Title :

Date :
Page No. :

```
printf(" Enter the root node:\n");
scanf("%d", &r);
DFS(r);
printf(" DFS Tree:\n");
for(i=1; i<=n; i++)
{   for(j=1; j<=n; j++)
      printf("\t %d", b[i][j]);
      printf("\n");
}
}

void DFS(int i)
{
    int j;
    visited[i]=1;
    for(j=1; j<=n; j++)
        if(!visited[j] && G[i][j]==1)
        {
            DFS(j);
            b[i][j]=1;
        }
}
```

Output:

Enter the no. of Vertices = 4

Enter the adjacency matrix of the graph:

0	1	0	1
1	0	1	0
0	1	0	1
1	0	1	0

Enter the root node : 1

Dfs tree :

0	1	0	0
0	0	1	0
0	0	0	1
0	0	0	0

21/9/19

procedure to solve dijkstra's Algorithm

→ given a graph G(V,E). where 'V' is the set of Vertices and 'E' is the set of Edges in the graph.

→ given a single vertex V_1 the problem is to find the shortest path from V_1 to every other node. Initially each vertex is assigned as unreachable and cost of the path is given as infinite to reach that vertex.

→ Cost of source vertex is given as 0

→ An empty set is filled by traversing each vertex and by calculating its shortest path from the source.

→ A set is completed on all the vertices are traversed.

Algorithm:

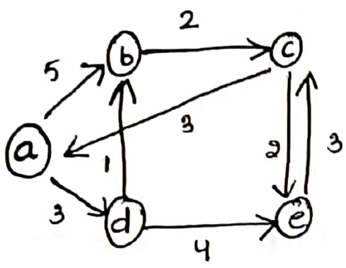
Step 1: Consider source vertex as 'a'. Distance of V is the cost of reach in vertex 'V' from 'a' parent of V is the the parent of node v and weight of [u,v] is the cost of traversing from u to v

Step 2: Initialize distance of 'V' is equal to infinite and ~~Parent of equal to~~ and distance dis(a)=0
v. par[v]=nil

Step 3: Select a node u for which weight of [a,u] is minimum.

Step 4: For each vertex 'V' connected with 'U': if dis[v] > dis[u] + weight [u,v] then dis[v] = dis[u] + weight [u,v] and par[v] = u

Step: 5 : Repeat Step 3 to 4 Until all Vertices are Traversed



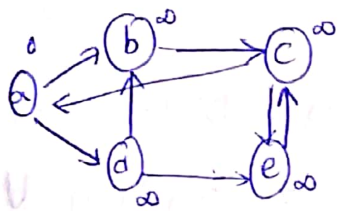
Weighted Matrix

A

	A	B	C	D	E
A	0	5	0	3	0
B	0	0	2	0	0
C	3	0	0	0	2
D	0	1	0	0	4
E	0	0	3	0	0

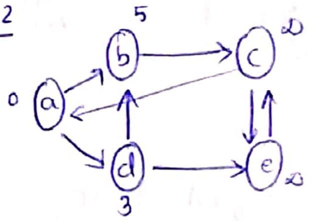
Step: 1

Source node = a



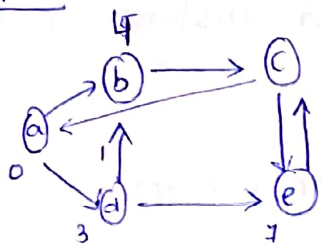
	a	b	c	d	e
source a	0	∞	∞	∞	∞

Step: 2



	a	b	c	d	e
a	0	~~∞~~	∞	~~∞~~	∞
a	0	5	∞	3	∞

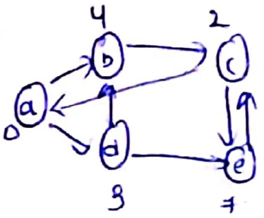
Step: 3



5 ≠ 4

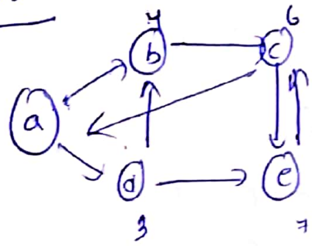
	a	b	c	d	e
s a	0	∞	∞	∞	∞
a	0	5	∞	3	∞
d	–	4 (a,d)	∞	3	7

Step 4.



	a	b	c	d	e
s a	0	∞	∞	∞	∞
a	0	5	∞	3	∞
d	-	4 (a,d)	∞	3	7 (a,d)
b	-	4 (a,d)	6 (a,b,d)	3	7 (a,d)

Step 5:

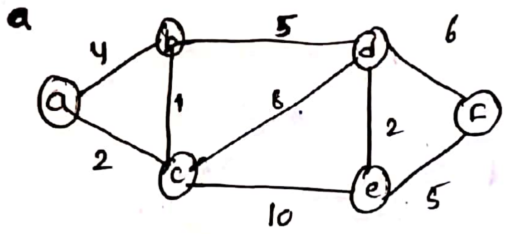


	a	b	c	d	e
s a	0	∞	∞	∞	∞
a	0	5	∞	3	∞
d	-	4 (a,d)	∞	3	7 (a,d)
b	-	4 (a,d)	6 (a,b,d)	3	7 (a,d)
c	-	4 (a,d)	6 (a,b,d)	3	7

Source

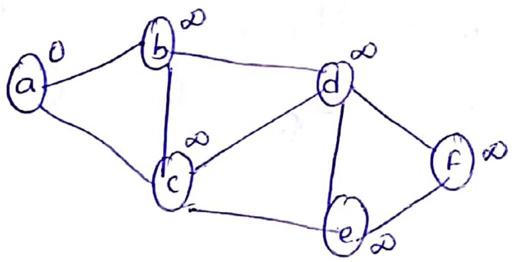
a-d	3
a-d-b-	4
a-d-b-c	6
a-d-e	7

(2)



$$
\begin{array}{c|cccccc}
 & A & B & C & D & E & F \\
\hline
A & 0 & 4 & 2 & 0 & 0 & 0 \\
B & 0 & 0 & 1 & 5 & 0 & 0 \\
C & 0 & 0 & 0 & 8 & 10 & 0 \\
D & 0 & 0 & 0 & 0 & 2 & 6 \\
E & 0 & 0 & 0 & 0 & 0 & 5 \\
F & 0 & 0 & 0 & 0 & 0 & 0
\end{array}
$$

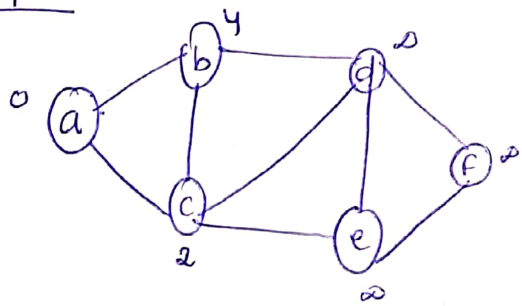
Step:1:



Sarce nod = a.

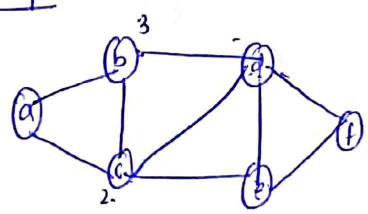
a	b	c	d	e	f
a 0	∞	∞	∞	∞	∞

Step:2



	a	b	c	d	e	f
(s) a	0	∞	∞	∞	∞	∞
a	0	4	2	∞	∞	∞

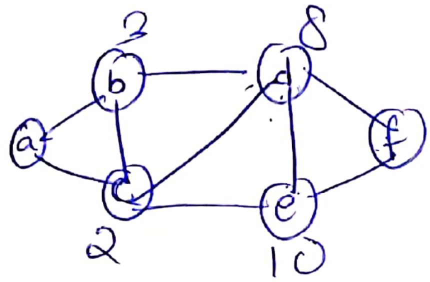
Step:3



	a	b	c	d	e	f
(s) a	0	∞	∞	∞	∞	∞
a	0	4	2	∞	∞	∞
c	-	3	2	10	12	∞

Scanned by CamScanner

Step-4 :



14	a	b	c	d	e	f
a	0	∞	∞	∞	∞	∞
a	0	4	2	∞	∞	∞
c	-	3	2	10	12	∞
b		3	2	8	12	∞
d	-	3	2	8	10	14

a - c 2

a - c - b 3

a - c - b - d - 8

a - c - b - d - e — 10

a - c - b - d - f - 14

27/9/19

Folding method:

The folding method was with the following steps:

Step 1 Divide the key value into a number of parts i.e Divide K into parts where Each part has the same number of digits Except the Last part which may have Lesser than the other part.

Step 2: Add the individual parts to get the hash value. If any Carry is Occured ignore.

Example:

34567 ∟100

34
56
7
——
97

100
321

32
1
——
33

5468

54
68
——
122

12
2
——
14

Collision:

→ If two key values want to share the same index value (or)

→ Two different keys produce the same location, then there is a possibility of two key values.

→ To Reduce or to remove the collision a method is used to solve this problem Collision resolution Techniques

→ The two most popular methods of resolutiong of collision are.

1) open addressing.

2) Chaining.

Collision resolution by open addressing:

In open addressing we follow the following Techniques.

1) Linear Probing
2) Quadratic probing
3) Rehashing probing.

Probing: The Process of Examining the memory location is called memory locations in the hash table is Called Probing. The Value which is used in probing is Called Probe.

CO - PO Attainment

VIDYA JYOTHI INSTITTUE OF TECHNOLOGY

DEPARTMENT OF INFORMATION TECHNOLOGY

Class: II year I sem | Course: DS | Batch: 2019-23

S.No	Reg.No	MID-1 Threshold 60%								MID-2 Threshold 60%								Threshold 60% (45M)
		Assignment1 (5M)	Theory (20M)	PART-B						Assignment2 (5M)	Theory (20M)	PART-B						
				M1- Q1 (2M) CO1	M1-Q2 (2M) CO2	M1- Q3 (2M) CO3	M1- Q4 (5M) CO1	M1-Q5 (5M) CO2	M1-Q6 (4M) CO3			M2- Q1 (2M) CO3	M2-Q2 (2M) CO4	M2- Q3 (2M) CO5	M2-Q4 (4M) CO3	M2-Q5 (5M) CO4	M2-Q6 (5M) CO5	End Exam (75M)
1	18911A1219	5	15	1	2	1	4	4	3	5	14	1	2	1	2	4	4	34
2	18911A1222	5	4	1	1	1	0	1	0	5	3	1	1	0	0	0	1	16
3	18911A1229	5	7	1	1	1	2	1	1	5	6	1	1	0	1	2	1	4
4	18911A1247	5	17	2	2	1	5	4	3	5	16	2	2	1	2	5	4	16
5	19911A1201	5	20	2	2	2	5	5	4	5	19	2	2	2	4	4	5	41
6	19911A1202	5	18	2	2	2	5	3	4	5	17	2	2	2	4	5	2	29
7	19911A1203	5	17	2	1	2	4	5	3	5	16	2	1	2	3	3	5	38
8	19911A1204	5	18	1	2	2	5	4	4	5	17	1	2	2	3	5	4	41
9	19911A1205	5	19	2	2	1	5	5	4	5	18	2	2	0	4	5	5	46
10	19911A1206	5	17	2	1	1	5	5	3	5	16	2	0	1	3	5	5	40
11	19911A1207	5	17	2	1	2	4	5	3	5	16	2	1	1	3	4	5	28
12	19911A1208	5	9	1	1	1	3	2	1	5	8	1	1	1	0	3	2	10
13	19911A1209	5	14	2	2	2	3	3	2	5	13	2	2	2	2	2	3	33
14	19911A1210	5	16	2	2	2	3	3	4	5	15	2	2	2	4	3	2	45
15	19911A1211	5	12	2	1	1	2	3	3	5	11	2	1	1	3	1	3	15
16	19911A1212	5	13	2	2	1	3	3	2	5	12	2	2	1	1	3	3	11
17	19911A1213	5	13	1	2	1	3	3	3	5	12	1	2	0	3	3	3	26
18	19911A1214	5	19	2	2	2	5	4	4	5	18	2	1	2	4	5	4	46
19	19911A1215	5	14	1	1	1	4	4	3	5	13	0	1	1	3	4	4	30
20	19911A1216	5	17	2	2	2	4	4	3	5	16	2	1	2	3	4	4	37
21	19911A1217	5	16	1	1	2	4	4	4	5	15	1	1	1	4	4	4	26
22	19911A1218	5	19	1	2	2	5	5	4	5	18	1	2	2	3	5	5	39
23	19911A1219	5	18	2	2	2	4	4	4	5	17	2	2	2	4	3	4	32
24	19911A1220	5	20	2	2	2	5	5	4	5	19	2	2	2	4	5	4	59
25	19911A1221	5	17	2	2	2	4	4	3	5	16	2	2	2	3	3	4	31
26	19911A1222	5	18	2	1	1	5	5	4	5	17	2	1	1	3	5	5	45
27	19911A1223	5	19	2	2	2	4	5	4	5	18	2	2	1	4	4	5	50
28	19911A1224	5	10	2	2	2	2	1	1	5	9	2	1	2	1	2	1	35
29	19911A1225	5	20	2	2	2	5	5	4	5	19	1	2	2	4	5	5	53
30	19911A1227	5	16	2	2	2	3	4	3	5	15	2	1	2	3	3	4	26
31	19911A1228	5	16	2	2	1	3	4	4	5	15	2	2	1	3	3	4	32

32	19911A1229	5	11	1	2	2	2	3	1	5	10	1	2	2	1	1	3	32
33	19911A1230	5	13	1	1	1	3	3	4	5	12	1	1	1	4	3	2	30
34	19911A1231	5	14	2	1	2	4	4	1	5	13	2	1	2	1	3	4	31
35	19911A1232	5	17	2	2	2	4	4	3	5	16	2	2	2	2	4	4	36
36	19911A1233	5	9	2	2	1	1	1	2	5	8	2	1	1	2	1	1	14
37	19911A1234	5	16	2	2	1	4	4	3	5	15	1	2	1	3	4	4	33
38	19911A1235	5	20	2	2	2	5	5	4	5	19	2	1	2	4	5	5	36
39	19911A1236	5	15	2	2	2	4	3	2	5	14	2	2	2	1	4	3	26
40	19911A1237	5	18	2	2	2	3	5	4	5	17	2	2	2	4	2	5	39
41	19911A1238	5	18	2	2	2	5	4	3	5	17	2	2	2	3	5	3	48
42	19911A1239	5	15	1	2	1	4	4	3	5	14	1	2	1	3	3	4	33
43	19911A1240	5	16	2	1	1	4	4	4	5	15	2	1	1	3	4	4	56
44	19911A1241	5	20	2	2	2	5	5	4	5	19	2	2	1	4	5	5	47
45	19911A1242	5	17	2	2	2	4	4	3	5	16	2	1	2	3	4	4	30
46	19911A1243	5	16	2	2	1	4	5	2	5	15	1	2	1	2	4	5	38
47	19911A1244	5	19	1	2	2	5	5	4	5	18	1	1	2	4	5	5	37
48	19911A1245	5	19	2	2	2	5	4	4	5	18	2	2	1	4	5	4	47
49	19911A1246	5	18	2	1	2	4	5	4	5	17	2	1	2	3	4	5	40
50	19911A1247	5	14	2	2	2	2	2	4	5	13	2	2	2	4	1	2	39
51	19911A1248	5	14	2	2	2	2	3	3	5	13	2	2	2	3	2	2	27
52	19911A1249	5	15	2	2	2	3	3	3	5	14	2	2	2	3	2	3	40
53	19911A1250	5	19	1	2	2	5	5	4	5	18	1	2	2	3	5	5	35
54	19911A1251	5	18	2	2	2	4	4	4	5	17	2	2	1	4	4	4	40
55	19911A1252	5	14	2	1	1	3	4	3	5	13	1	1	1	3	3	4	16
56	19911A1253	5	15	2	1	2	3	3	4	5	14	2	1	1	4	3	3	33
57	19911A1254	5	15	2	2	2	4	3	2	5	14	2	2	2	1	4	3	26
58	19911A1255	5	11	2	1	2	2	2	2	5	10	2	1	2	2	1	2	26
59	19911A1256	5	17	2	2	2	4	4	3	5	16	2	2	2	3	4	3	37
60	19911A1257	5	7	2	1	1	1	1	1	5	6	2	1	1	0	1	1	28
61	19911A1258	5	16	2	2	1	4	4	3	5	15	2	1	1	3	4	4	26
62	19911A1259	5	17	1	2	2	4	4	4	5	16	1	1	2	4	4	4	26
63	19911A1260	5	17	2	2	1	5	4	3	5	16	2	2	1	2	5	4	26
64	19911A1261	5	15	2	2	2	3	4	2	5	14	2	2	2	2	2	4	26
65	19911A1262	5	0	0	0	0	0	0	0	5	-1	0	0	0	0	0	0	6
66	19911A1263	5	14	1	1	1	3	4	4	5	13	1	1	1	4	2	4	26
67	19911A1264	5	17	2	1	2	4	5	3	5	16	2	1	2	2	4	5	48
68	19911A1265	5	9	2	2	1	2	1	1	5	8	2	1	1	1	2	1	26
69	19911A1266	5	15	2	2	2	4	2	3	5	14	1	2	2	3	4	2	26

70	19911A1267	5	17	1	2	2	4	4	4	5	16	1	1	2	4	4	4	26
71	19911A1268	5	15	2	2	1	2	5	3	5	14	2	1	1	3	2	5	26
72	19911A1269	5	13	2	2	2	3	2	2	5	12	2	2	1	2	3	2	20
73	19911A1270	5	20	2	2	2	5	5	4	5	19	2	2	2	3	5	5	37
74	19911A1271	5	17	1	2	2	5	4	3	5	16	1	2	2	3	4	4	36
75	19911A1272	5	16	2	2	2	4	3	3	5	15	2	2	2	3	4	2	44
76	19911A1273	5	17	2	2	2	4	4	3	5	16	2	2	2	3	3	4	39
77	19911A1274	5	11	2	2	1	2	2	2	5	10	2	2	1	1	2	2	26
78	19911A1275	5	**12**	2	1	2	2	2	3	5	**11**	2	1	1	3	2	2	12
79	19911A1276	5	17	2	2	2	3	5	3	5	16	2	1	2	3	3	5	45
80	19911A1277	5	15	2	2	2	3	2	4	5	14	1	2	2	4	3	2	41
81	19911A1278	5	18	2	2	2	5	3	4	5	17	2	1	2	4	5	3	44
82	19911A1279	5	12	1	2	2	3	2	2	5	11	1	2	1	2	3	2	14
83	19911A1280	5	19	1	2	2	5	5	4	5	18	1	2	2	3	5	5	44
84	19911A1281	5	18	2	2	1	5	4	4	5	17	2	2	1	4	4	4	28
85	19911A1282	5	11	1	1	1	3	3	2	5	10	1	1	1	2	3	2	14
86	19911A1283	5	18	2	1	2	5	4	4	5	17	2	1	2	4	4	4	35
87	19911A1284	5	16	1	1	2	4	4	4	5	15	1	1	2	3	4	4	32
88	19911A1285	5	14	1	1	2	2	4	4	5	13	1	1	1	4	2	4	18
89	19911A1286	5	17	2	2	2	3	4	4	5	16	2	1	2	4	3	4	37
90	19911A1287	5	18	2	2	2	4	5	3	5	17	1	2	2	3	4	5	33
91	19911A1288	5	15	2	2	2	2	3	4	5	14	2	1	2	4	2	3	26
92	19911A1289	5	20	2	2	2	5	5	4	5	19	2	2	1	4	5	5	46
93	19911A1290	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	A
94	19911A1291	5	18	2	2	2	4	5	3	5	17	2	2	2	2	4	5	26
95	19911A1292	5	12	2	2	2	4	1	1	5	11	2	2	2	1	3	1	26
96	19911A1293	5	17	1	2	2	5	4	3	5	16	1	2	2	2	5	4	26
97	19911A1294	5	19	2	1	2	5	5	4	5	18	2	1	1	4	5	5	46
98	19911A1295	5	20	2	2	2	5	5	4	5	19	2	1	2	4	5	5	44
99	19911A1296	5	18	2	2	2	5	5	2	5	17	1	2	2	2	5	5	41
100	19911A1297	5	13	2	2	2	2	2	3	5	12	2	1	2	3	2	2	32
101	19911A1298	5	10	2	1	1	3	2	1	5	9	2	1	1	1	2	2	33
102	19911A1299	5	15	2	2	2	4	3	2	5	14	2	2	2	2	3	3	40
103	19911A12A0	5	18	2	2	2	3	5	4	5	17	2	2	2	3	3	5	35
104	19911A12A1	5	12	1	2	2	2	2	3	5	11	1	2	1	3	2	2	6
105	19911A12A2	5	17	2	2	2	4	4	3	5	16	2	1	2	3	4	4	27
106	19911A12A3	5	18	2	2	2	4	4	4	5	17	1	2	2	4	4	4	39
107	19911A12A4	5	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	A

108	19911A12A5	5	15	2	1	2	4	3	3	5	14	1	1	2	3	4	3	26
109	19911A12A6	5	20	2	2	2	5	5	4	5	19	2	1	2	4	5	5	47
110	19911A12A7	5	7	1	1	2	1	1	1	5	6	1	1	1	1	1	1	2
111	19911A12A8	5	20	2	2	2	5	5	4	5	19	2	2	2	3	5	5	39
112	19911A12A9	5	13	2	2	1	4	3	1	5	12	2	2	1	1	3	3	26
113	19911A12B0	5	16	2	1	1	4	5	3	5	15	2	1	1	3	4	4	37
114	19911A12B1	5	19	1	2	2	5	5	4	5	18	1	2	2	4	4	5	37
115	19911A12B2	5	16	2	2	2	5	3	2	5	15	2	2	2	1	5	3	41
116	19911A12B3	5	16	2	2	2	5	3	2	5	15	2	2	1	2	5	3	38
117	19911A12B4	5	15	2	2	2	2	3	4	5	14	2	1	2	4	2	3	26
118	19911A12B5	5	16	2	2	2	5	3	2	5	15	1	2	2	2	5	3	43
119	19911A12B6	5	18	2	1	1	5	5	4	5	17	2	1	1	3	5	5	31
120	19911A12B7	5	16	2	2	2	5	3	2	5	15	2	2	2	2	4	3	30
121	19911A12B8	5	17	2	1	2	5	4	3	5	16	2	1	2	3	5	3	38
122	19911A12B9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	A
123	19911A12C1	5	20	2	2	2	5	5	4	5	19	2	2	2	4	4	5	38
124	19911A12C2	5	19	2	1	2	5	5	4	5	18	2	1	2	3	5	5	47
125	19911A12C3	5	19	2	2	1	5	5	4	5	18	2	1	1	4	5	5	27
126	19911A12C4	5	13	2	2	2	2	3	2	5	12	1	2	2	2	2	3	32
127	19911A12C5	5	18	2	1	2	5	4	4	5	17	2	1	1	4	5	4	17
128	19911A12C6	5	20	2	2	2	5	5	4	5	19	2	2	2	3	5	5	50
129	19911A12C7	5	20	2	2	2	5	5	4	5	19	2	2	2	4	4	5	50
130	19911A12C8	5	19	2	2	2	4	5	4	5	18	2	2	2	4	4	4	54
131	19911A12C9	5	16	2	2	2	4	3	3	5	15	2	2	2	3	3	3	26
132	19911A12D0	5	20	2	2	2	5	5	4	5	19	2	2	2	3	5	5	50
133	19911A12D1	5	19	1	2	2	5	5	4	5	18	1	2	1	4	5	5	52
134	19911A12D2	5	18	2	2	2	3	5	4	5	17	2	1	2	4	3	5	44
135	19911A12D3	5	17	2	2	2	4	3	4	5	16	1	2	2	4	4	3	16
136	19911A12D4	5	18	1	1	2	5	5	4	5	17	1	1	1	4	5	5	39
137	19911A12D5	5	17	2	2	2	4	4	3	5	16	2	2	2	2	4	4	31
138	19911A12D6	5	17	2	1	2	5	4	3	5	16	2	1	2	3	4	4	26
139	19911A12D7	5	20	2	2	2	5	5	4	5	19	2	2	2	4	5	4	31
140	19911A12D8	5	18	2	2	2	5	4	3	5	17	2	2	2	3	4	4	32
141	19911A12D9	5	17	2	2	2	4	4	3	5	16	2	2	2	2	4	4	28
142	19911A12E0	5	19	2	2	2	5	4	4	5	18	2	2	1	4	5	4	36
143	19911A12E1	5	16	2	1	1	4	5	3	5	15	1	1	1	3	4	5	28
144	19911A12E2	5	19	1	2	2	5	5	4	5	18	1	1	2	4	5	5	26
145	19911A12E3	5	20	2	2	2	5	5	4	5	19	2	2	1	4	5	5	30

146	19911A12E4	5	17	2	2	2	3	4	4	5	16	2	2	2	3	3	4	26
147	19911A12E5	5	20	2	2	2	5	5	4	5	19	2	2	2	4	4	5	40
148	19911A12E6	5	20	2	2	2	5	5	4	5	19	2	2	2	4	5	4	36
149	19911A12E7	5	18	2	2	2	5	3	4	5	17	2	2	2	4	4	3	40
150	19911A12E8	5	17	2	1	2	5	4	3	5	16	2	1	2	2	5	4	35
151	19911A12E9	5	18	2	2	2	4	4	4	5	17	2	2	1	4	4	4	48
152	19911A12F0	5	14	2	2	2	4	2	2	5	13	2	1	2	2	4	2	33
153	19911A12F1	5	18	1	2	2	5	4	4	5	17	1	1	2	4	5	4	43
154	19911A12F2	5	18	2	2	2	4	4	4	5	17	2	2	1	4	4	4	46
155	19911A12F3	5	19	2	1	2	5	5	4	5	18	2	1	2	3	5	5	37
156	19911A12F4	5	18	2	2	2	5	3	4	5	17	2	2	2	4	4	3	39
157	19911A12F5	5	18	2	1	2	5	4	4	5	17	2	1	2	4	5	3	35
158	19911A12F6	5	19	2	2	2	4	5	4	5	18	2	2	2	4	3	5	37
159	19911A12F7	5	19	2	2	2	4	5	4	5	18	2	2	2	3	4	5	29
160	19911A12F8	5	18	2	2	2	5	4	3	5	17	2	2	1	3	5	4	30
161	19911A12F9	5	20	2	2	2	5	5	4	5	19	2	1	2	4	5	5	32
162	19911A12G0	5	19	1	2	2	5	5	4	5	18	1	1	2	4	5	5	40
163	19911A12G1	5	16	2	2	2	3	3	4	5	15	2	2	1	4	3	3	42
164	19911A12G2	5	18	2	2	2	5	3	4	5	17	2	2	2	3	5	3	35
165	19911A12G3	5	19	2	1	2	5	5	4	5	18	2	1	2	4	4	5	38
166	19911A12G4	5	20	2	2	2	5	5	4	5	19	2	2	2	4	5	4	47
167	19911A12G5	5	17	2	2	2	4	4	3	5	16	2	2	2	3	3	4	50
168	19911A12G6	5	18	2	2	2	4	4	4	5	17	2	2	2	3	4	4	31
169	19911A12G7	5	19	2	2	1	5	5	4	5	18	2	2	0	4	5	5	40
170	19911A12G8	5	20	2	2	2	5	5	4	5	19	2	1	2	4	5	5	41
171	19911A12G9	5	17	2	2	2	4	3	4	5	16	1	2	2	4	4	3	36
172	19911A12H0	5	17	2	1	2	5	4	3	5	16	2	1	1	3	5	4	29
173	19911A12H1	5	19	2	2	1	5	5	4	5	18	2	2	1	3	5	5	42
174	19911A12H2	5	14	1	2	2	3	5	1	5	13	1	2	2	1	2	5	26
175	19911A12H3	5	-5	0	0	0	0	0	0	5	-6	0	0	0	0	0	0	12
176	19911A12H4	5	19	2	1	2	5	5	4	5	18	2	1	2	4	5	4	38
177	19911A12H5	5	20	2	2	2	5	5	4	5	19	2	2	2	4	4	5	42
178	19911A12H6	5	17	2	2	2	4	4	3	5	16	2	2	2	2	4	4	16
179	20915A1201	5	17	2	1	2	4	5	3	5	16	2	1	1	3	4	5	26
180	20915A1202	5	15	2	2	2	4	4	1	5	14	2	1	2	1	4	4	16
181	20915A1203	5	15	2	2	2	4	3	2	5	14	1	2	2	2	4	3	29
182	20915A1204	5	16	2	2	2	4	4	2	5	15	2	1	2	2	4	4	26
183	20915A1205	5	19	2	2	2	5	4	4	5	18	2	2	1	4	5	4	42

184	20915A1206	5	18	2	2	[illegible]	5	4	3	5	17	2	2	2	2	5	4	36
185	20915A1207	5	14	2	2	1	3	3	3	5	13	2	2	1	3	2	3	28
186	20915A1208	5	17	2	2	2	5	4	2	5	16	2	2	2	2	5	3	29
187	20915A1209	5	17	2	2	2	4	5	2	5	16	2	2	2	2	3	5	26
188	20915A1210	5	16	1	1	1	5	5	3	5	15	1	1	1	2	5	5	44
189	20915A1211	5	16	2	1	2	5	3	3	5	15	2	1	1	3	5	3	43
190	20915A1212	5	18	2	2	1	5	5	3	5	17	2	1	1	3	5	5	27
191	20915A1213	5	13	1	1	1	4	4	2	5	12	1	1	1	1	4	4	11
192	20915A1214	5	19	2	1	2	5	5	4	5	18	2	1	2	4	4	5	26
Average marks		4.9	16.0	1.8	1.7	1.7	3.9	3.8	3.1	4.9	15.0	1.7	1.5	1.6	2.9	3.7	3.7	33.1
No of students attemped		192	192	192	192	192	192	192	192	192	192	192	192	192	192	192	192	189
%of students scored 60%		98.96	97.40	0.00	0.00	0.00	86.46	86.46	77.08	98.96	97.40	0.00	0.00	0.00	70.31	82.29	83.33	99.47
CO ATTAINMENT LEVEL		3.0	3.0	0.0	0.0	0.0	3.0	3.0	2.0	3.0	3.0	0.0	0.0	0.0	2.0	3.0	3.0	3.0

ASSESSMENT OF COs FOR THE COURSE							
COs	Method	value	CO Attainment	Assignments	CO Attainment (Internal - Theory)	CO Attainment (End Exam)	Overall CO Attainment
CO1	MI Q1	0.0	1.5	3.0	1.4	3.00	2.80
	MI Q5	3.0					
CO2	MI Q2	0.0	1.5				
	MI Q6	3.0					
CO3	MI Q3	0.0	1.0				
	MI Q7	2.0					
	M2 Q1	0.0					
	M2 Q4	2.0					
CO4	M2 Q2	0.0	1.5				
	M2 Q5	3.0					

CO5	M2 Q3	0.0	1.5				
	M2 Q6	3.0					

S

Course End Survey Form

[redacted]

1. Are you able to analyze the representation of various data structures and implement the mechanisms of Stacks and Queues with their applications.46 responses

Slight 1
Moderate 2
Substantial 43

2. Are you able to implement the operations like searching, insertion, deletions and traversing mechanisms on Binary Trees? 46 responses

Slight 1
Moderate 3
Substantial 42

3. Are you able to implement various advance concepts of trees with real time applications? 46 responses

Slight 2
Moderate 4
Substantial 40

4. Are you able to implement various algorithms on graph data structures, including finding the minimum spanning tree, shortest path with real time applications, etc? 46 responses

Slight 1
Moderate 3
Substantial 42

5. Are you able to outline the concepts of hashing, collision and its resolution methods using hash function? 46 responses

Slight 2
Moderate 1
Substantial 43