Vidya Jyothi Institute of Technology

(An Autonomous Institution)
(Accredited by NAAC & NBA, Approved by AICTE New Delhi & Permanently Affiliated to JNTUH)
Aziznagar Gate, C.B. Post, Hyderabad-500 075

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

REGULATION: R19

BATCH: 2019-23

ACADEMIC YEAR: 2020-2021

PROGRAM: B.TECH (COMPUTER SCIENCE AND ENGINEERING)

YEAR/SEM: II/I

COURSE NAME: DS and PP Lab

COURSE CODE: A40506

COURSE COORDINATOR: V.Srilakshmi

Head of the Department
Computer Science and Engineering
VJIT, Hyderabad-50075.

LABORATORY COURSE FILE INDEX

S.No	Contents
1	List of Experiments
2	Laboratory Course Outcomes
3	Program Outcomes (PO) , Program Specific Outcomes (PSO)& Program Educational Objectives (PEO)
4	Mapping of Course Outcomes (CO's) with Program Outcomes (PO's) & Program Specific Outcomes (PSO's)
5	Lab Manual
6	Assessment Sheets
7	Course End Survey Form

VIDYA JYOTHI INSTITUTE OF TECHNOLOGY
Department of Computer Science and Engineering

LIST OF EXPERIMENTS

S.no	EXPERIMENTS
1	Week 1 Installation and Environment setup of python. Write a program to illustrate string built in functions
2	Week 2 Write a program to evaluate postfix notations Write a program to demonstrate the use of basic Data Types c) Write a program to demonstrate the Operators and Expressions
3	Week 3 Write a program to convert infix to post fix notation Write a program to demonstrate the Functions and parameter passing Techniques
4	Week 4 Write a program to illustrate tree traversals a)In order b)Preorder c)Post order
5	Week 5) Write a Program to implement **i.** Packages**ii.**Modules **iii.** Built-in Functions b) Write a Program to implement **i.** List **ii.**Tuple **iii.** Dictionaries
6	Week 6 Write a program to illustrate insertion, deletion and searching in Binary Search Tree. Programs on Stings, String Operations and Regular Expressions

	Write a Program to implement Class and Object
7	Week 7 Write a program to illustrate Graph traversals a) · Breadth First Search Depth First Search
8	Week 8 Write a Program to implement Static and Instance methods, Abstract Classes and Interfaces. Write a program to compute distance between two points taking input from the user (Pythagorean Theorem) b) Write a program to convert a given decimal number to other base systems
9	Week 9 Write a program to illustrate Insertion, deletion and Rotation on AVL Trees.
10	Week 10 Write a program to implement Inheritance
11	Week 11 Write a program to implement Polymorphism
12	Week 12 Write a program to implement Files Write a program to Exception Handling.

VIDYA JYOTHI INSTITUTE OF TECHNOLOGY
Department of Computer Science and Engineering

LABORATORY COURSE OUTCOMES

Course name : DS and PP Lab

Academic Year: 2020- 2021

Regulation: R19

Course Outcomes (COs):

At the end of the course the student should be able to:

CO1: Develop the programs on stacks, trees and its applications

CO2: Design and implement of programs on BST and Graph Traversals

CO3: Apply hashing techniques in real world applications

CO4: Implement oops concepts in Python

CO5: Develop programs and modules and packages

CO6: Design programs that handle errors

VIDYA JYOTHI INSTITUTE OF TECHNOLOGY
Department of Computer Science and Engineering

Program Outcomes(po's) ,PSO's,CO's

Programme Outcomes (PO's)

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization for the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, 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 public health and safety, and 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 5odeling 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 the 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.

Program Specific Objectives (PSO's):

PSO 1: The ability to design and develop Algorithms to provide optimized solutions for societal needs

PSO 2: Apply standard approaches and practices in Software Project Development through trending technologies

Vidya Jyothi Institute of Technology

(An Autonomous Institution)
(Accredited by NAAC & NBA, Approved by AICTE New Delhi & Permanently Affiliated to JNTUH)
Aziznagar Gate, C.B. Post, Hyderabad-500 075

Mapping Of Course Outcomes (Co's) With Program Outcomes (Po's) & Program Specific Outcomes (Pso's)

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	3	3	3	3	3	3	3	3	3	3	3	3
CO2	3	3	3	3	3	3	3	3	3	3	3	3
CO3	3	3	3	3	3	3	3	3	3	3	3	3
CO4	3	3	3	3	3	3	3	3	3	3	3	3
CO5	3	3	3	3	3	3	3	3	3	3	3	3
CO6	3	3	3	3	3	3	3	3	3	3	3	3
AVG	3	3	3	3	3	3	3	3	3	3	3	3

	PSO1	PSO2
CO1	3	3
CO2	3	3
CO3	3	3
CO4	3	3
CO5	3	3
CO6	3	3
AVG	3	3

Course Coordinator

HOD

Head of the Department
Computer Science and Engineering
VJIT, Hyderabad-500075.

LABORATORY MANUAL

VIDYA JYOTHI INSTITUTE OF TECHNOLOGY

(An Autonomous Institution)

(Accredited by NAAC, NBA, Approved by AICTE New Delhi & Permanently Affiliated to JNTUH)

Aziznagar Gate, C.B. Post, Hyderabad-500 075

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Continuous Lab Assessment sheet

Academic year: 2020-21 Regulation: R18 Year/sem/ Section: II/I/C

Course name: Data Structures + Python Course code: A23506 Date:

S.NO	HNO	Week1	Week2	Week3	Week4	Week5	Week6	Week7	Week8	Week9	Week10	Week11	Week12	Week13	Week14	Week15	Week16	Reduced to 15	Intenal 10	Total 25
1	199114405C1	A	A	A	A	A	A											10	8	18
2	5C2	A	A	A	A	A	A											10	8	18
3	5C3	23	25	25	25	24	23											13	10	23
4	5C4	A	A	A	A	A	A											10	8	18
5	5C5	22	24	25	25	23	24											10	10	20
6	5C6	25	25	21	24	22	23											15	10	25
7	5C7	A	A	A	A	A	A											10	7	17
8	5C8	24	23	24	23	23	23											13	10	23
9	5C9	22	21	23	21	21	22											10	10	20
10	5D0	23	25	25	25	23	21											12	10	22
11	5D1	22	24	21	24	24	24											11	10	21
12	5D2	21	22	22	23	23	25											15	10	25
13	5D3	25	25	22	22	22	22											10	7	17
14	5D4	24	A	A	24	21	21											10	10	20
15	5D5	24	A	A	A	A	A											10	9	19
16	5D6	A	A	A	25	23	25											10	6	16
17	5D7	24	24	25	24	24	22											15	10	25
18	5D8	A	24	24	A	A	A											12	10	22
19	5D9	A	23	24	23	25	A											10	9	19
20	5E0	23	21	23	25	25	23											15	10	25
21	5E1	25	24	23	21	24	24											15	10	25
22	5E2	23	23	22	22	23	21											10	8	18
23	5E3	22	25	21	22	22	A											10	8	18
24	5E4	24	24	25	23	25	A											11	10	21
25	5E5	A	A	A	22	24	A											10	9	19
26	5E6	21	A	A	A	A	22											13	10	23
27	5E7	24	24	25	21	22	25											15	10	25
28	5E8	A	A	A	25	21	24											10	10	20
29	5E9	23	23	24	A	A	A											10	8	18
30	5F0	22	25	25	24	24	24											10	10	20

S.NO	HNO	Week1	Week2	Week3	Week4	Week5	Week6	Week7	Week8	Week9	Week10	Week11	Week12	Week13	Week14	Week15	Week16	Reduced to	Intenal	Total 25
31	199114055F1	25	24	25	25	23	23											14	10	24
32	5F2	25	21	22	23	24	24											11	10	21
33	5F3	A	A	A	A	A	A											11	10	21
34	6F4	23	A	A	21	23	21											10	10	20
35	5F5	24	25	21	A	A	22											10	9	19
36	5F6	23	24	24	A	A	24											12	10	22
37	5F7	24	A	A	A	A	A											10	8	18
38	5F8	25	A	A	A	A	23											10	9	19
39	5F9	24	A	A	A	A	22											14	10	24
40	5G0	21	21	24	21	22	25											15	10	25
41	5G1	23	24	25	25	21	24											12	10	22
42	5G2	A	A	A	A	A	A											0	0	00
43	5G3	24	21	23	25	25	22											11	10	21
44	5G4	A	22	25	25	24	25											10	10	20
45	5G5	25	25	24	24	23	24											11	10	21
46	5G6	21	23	21	24	22	23											15	10	25
47	5G7	22	A	A	A	A	A											10	9	19
48	5G8	A	A	A	A	A	A											10	8	18
49	5G9	A	24	24	23	21	21											10	10	20
50	5H0	23	24	24	23	25	24											15	10	25
51	5H1	22	22	23	25	25	23											15	10	25
52	5H2	21	A	A	23	24	25											10	7	17
53	5H3	A	23	24	22	23	22											11	10	21
54	5H4	A	A	A	A	A	24											15	10	25
55	5H5	22	22	22	25	23	22											10	9	19
56	5H6	21	22	22	24	24	24											10	8	18
57	5H7	A	21	23	23	23	25											11	10	21
58	5H8	23	23	21	24	24	22											11	10	21
59	5H9	24	A	A	A	A	23											12	10	22
60	5I0	22	22	25	23	21	21											12	10	22
61	169114055B	A	A	A	A	A	A											0	0	00
62	5B5	A	A	A	A	A	25											10	10	20
63																				
64																				
65																				
66																				
67																				
68																				

Course Instructor

HOD

VIDYA JYOTHI INSTITUTE OF TECHNOLOGY

(An Autonomous Institution)

(Accredited by NAAC, NBA, Approved by AICTE New Delhi & Permanently Affiliated to JNTUH)

Aziznagar Gate, C.B. Post, Hyderabad-500 075

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Continuous Lab Assessment sheet

Academic year: 2020-21 Regulation: R18

Course name: Data Structures / Python lab Course code: 4CS05

Year/sem: II-I Section: C Date: 21/12/2020

S. No	Ht.No	Observation (5M)	Completion of Experiment (10M)	Viva (5M)	Record (5M)	Total (25M)
1.	1991/4 05C1			AB		
2.	5C2			AB		
3.	5C3	5	10	5	5	25
4.	5C4			AB		
5.	5C5	5	10	4	5	24
6.	5C6	5	10	5	5	25
7.	5C7			AB		
8.	5C8	5	10	3	5	23
9.	5C9	5	10	1	5	21
10.	5D0	5	10	5	5	25
11.	5D1	5	10	4	5	24
12.	5D2	5	10	2	5	22
13.	5D3	5	10	5	5	25
14.	5D4			AB		
15.	5D5			AB		
16.	5D6			AB		
17.	5D7	5	10	4	5	24
18.	5D8	5	10	4	5	24
19.	5D9	5	10	3	5	23
20.	5E0	5	10	1	5	21
21.	5E1	5	10	4	5	24
22.	5E2	5	10	3	5	23
23.	5E3	5	10	5	5	25
24.	5E4	5	10	4	5	24
25.	5E5			AB		
26.	5E6			AB		
27.	5E7	5	10	4	5	24
28.	5E8			AB		
29.	5E9	5	10	3	5	23

30.	1491140 5F0	5	10	5	5	25
31.	5F1	5	10	4	5	24
32.	5F2	5	10	1	5	21
33.	5F3	—	—	AB	—	—
34.	5F4	—	—	AB	—	—
35.	5F5	5	10	5	5	25
36.	5F6	5	10	4	5	24
37.	5F7	—	—	AB	—	—
38.	5F8	—	—	AB	—	—
39.	5F9	—	—	AB	—	—
40.	5G0	5	10	2	5	22
41.	5G1	5	10	5	5	25
42.	5G2	—	—	AB	—	—
43.	5G3	5	10	1	5	21
44.	5G4	5	10	2	5	22
45.	5G5	5	10	5	5	25
46.	5G6	5	10	3	5	23
47.	5G7	—	—	AB	—	—
48.	5G8	—	—	AB	—	—
49.	5G9	5	10	4	5	24
50.	5H0	5	10	4	5	24
51.	5H1	5	10	2	5	22
52.	5H2	—	—	AB	—	—
53.	5H3	5	10	3	5	23
54.	5H4	—	—	AB	—	—
55.	5H5	5	10	2	5	22
56.	5H6	5	10	2	5	22
57.	5H7	5	10	1	5	21
58.	5H8	5	10	3	5	23
59.	5H9	—	—	AB	—	—
60.	5J0	5	10	2	5	22
61.	1891140 5J5	—	—	AB	—	—
62.	1891140 5J6	—	—	AB	—	—
63.						
64.						
65.						

Note: At the end of the semester total marks are calculated to 15.

(Course Instructor)

VIDYA JYOTHI INSTITUTE OF TECHNOLOGY

(An Autonomous Institution)

(Accredited by NAAC, NBA, Approved by AICTE New Delhi & Permanently Affiliated to JNTUH)

Aziznagar Gate, C.B. Post, Hyderabad-500 075

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Continuous Lab Assessment sheet

Academic year: 2020-21 Regulation: R18

Course name: Data Structures & Pythonlab Course code: A23506

Year/sem: II-I Section: C Date: 21/12/2020

S. No	Ht.No	Observation (5M)	Completion of Experiment (10M)	Viva (5M)	Record (5M)	Total (25M)
1.	1991A05C1	—	—	AB	—	—
2.	5C2	—	—	AB	—	—
3.	5C3	5	10	5	5	25
4.	5C4	—	—	AB	—	—
5.	5C5	5	10	5	5	25
6.	5C6	5	10	1	5	21
7.	5C7	—	—	AB	—	—
8.	5C8	5	10	4	5	24
9.	5C9	5	10	3	5	23
10.	5D0	5	10	5	5	25
11.	5D1	5	10	1	5	21
12.	5D2	5	10	2	5	22
13.	5D3	5	10	2	5	22
14.	5D4	—	—	AB	—	—
15.	5D5	—	—	AB	—	—
16.	5D6	—	—	AB	—	—
17.	5D7	5	10	5	5	25
18.	5D8	5	10	4	5	24
19.	5D9	5	10	4	5	24
20.	5E0	5	10	3	5	23
21.	5E1	5	10	3	5	23
22.	5E2	5	10	2	5	22
23.	5E3	5	10	1	5	21
24.	5E4	5	10	A 5	5	25
25.	5E5	—	—	AB	—	—
26.	5E6	—	—	AB	—	—
27.	5E7	5	10	5	5	25
28.	5E8	—	—	AB	—	—
29.	5E9	5	10	4	5	24

30.	1991A05F0	5	10	5	5	25
31.	5F1	5	10	5	5	25
32.	5F2	5	10	2	5	22
33.	5F3	—	—	AB	—	—
34.	5F4	—	—	AB	—	—
35.	5F5	5	10	1	5	21
36.	5F6	5	10	M	5	24
37.	5F7	—	—	AB	—	—
38.	5F8	—	—	AB	—	—
39.	5F9	—	—	AB	—	—
40.	5G0	5	10	M	5	24
41.	5G1	5	10	5	5	25
42.	5G2	—	—	AB	—	—
43.	5G3	5	10	3	5	23
44.	5G4	5	10	5	5	25
45.	5G5	5	10	2	5	22
46.	5G6	5	10	1	5	21
47.	5G7	—	—	AB	—	—
48.	5G8	—	—	AB	—	—
49.	5G9	5	10	1	5	21
50.	5H0	5	10	M	5	24
51.	5H1	5	10	3	5	23
52.	5H2	—	—	AB	—	—
53.	5H3	5	10	M	5	24
54.	5H4	—	—	AB	—	—
55.	5H5	5	10	2	5	22
56.	5H6	5	10	2	5	22
57.	5H7	5	10	3	5	23
58.	5H8	5	10	1	5	21
59.	5H9	—	—	AB	—	—
60.	5J0	5	10	5	5	25
61.	1891A05B5	—	—	AB	—	—
62.	1891A05J5	—	—	AB	—	—
63.						
64.						
65.						

Note: At the end of the semester total marks are calculated to 15.

(Course Instructor)

VIDYA JYOTHI INSTITUTE OF TECHNOLOGY

(An Autonomous Institution)

(Accredited by NAAC, NBA, Approved by AICTE New Delhi & Permanently Affiliated to JNTUH)

Aziznagar Gate, C.B. Post, Hyderabad-500 075

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Continuous Lab Assessment sheet

Academic year: 2020-21 Regulation: R18

Course name: Data structures Python Course code: A73506

Year/sem: II-I Section: C Date: 16/12/2020

S. No	Ht.No	Observation (5M)	Completion of Experiment (10M)	Viva (5M)	Record (5M)	Total (25M)
1.	19911A05C1	—	—	A	—	—
2.	5C2	—	—	A	—	—
3.	5C3	5	10	5	5	25
4.	5C4	—	—	A	—	—
5.	5C5	5	10	5	5	25
6.	5C6	5	10	M	5	24
7.	5C7	—	—	A	—	—
8.	5C8	5	10	3	5	23
9.	5C9	5	10	1	5	21
10.	5D0	5	10	5	5	25
11.	5D1	5	10	M	5	24
12.	5D2	5	10	3	5	23
13.	5D3	5	10	2	5	22
14.	5D4	5	10	M	5	24
15.	5D5	—	—	A	—	—
16.	5D6	5	10	5	5	25
17.	5D7	5	10	M	5	24
18.	5D8	—	—	A	—	—
19.	5D9	5	10	3	5	23
20.	5E0	5	10	5	5	25
21.	5E1	5	10	1	5	21
22.	5E2	5	10	2	5	22
23.	5E3	5	10	2	5	22
24.	5E4	5	10	3	5	23
25.	5E5	5	10	2	5	22
26.	5E6	—	—	A	—	—
27.	5E7	5	10	1	5	21
28.	5E8	5	10	5	5	25
29.	5E9	—	—	A	—	—

30.	1941405F0	5	10	4	5	24
31.	5F1	5	10	5	5	25
32.	5F2	5	10	3	5	23
33.	5F3	—	A	—	—	
34.	5F4	5	10	1	5	21
35.	5F5	—	A	—	—	
36.	5F6	—	A	—	—	
37.	5F7	—	A	—	—	
38.	5F8	—	A	—	—	
39.	5F9	—	A	—	—	
40.	5G0	5	10	1	5	21
41.	5G1	5	10	5	5	25
42.	5G2	—	A	—	—	
43.	5G3	5	10	5	5	25
44.	5G4	5	10	5	5	25
45.	5G5	5	10	4	5	24
46.	5G6	5	10	4	5	24
47.	5G7	—	A	—	—	
48.	5G8	—	A	—	—	
49.	5G9	5	10	3	5	23
50.	5H0	5	10	3	5	23
51.	5H1	5	10	5	5	25
52.	5H2	5	10	3	5	23
53.	5H3	5	10	2	5	22
54.	5H4	—	A	—	—	
55.	5H5	5	10	5	5	25
56.	5H6	5	10	4	5	24
57.	5H7	5	10	3	5	23
58.	5H8	5	10	4	5	24
59.	5H9	—	A	—	—	
60.	520	5	10	3	5	23
61.	1891140558	—	A	—	—	
62.	1891140585	—	A	—	—	
63.						
64.						
65.						

Note: At the end of the semester total marks are calculated to 15.

(Course Instructor)

VIDYA JYOTHI INSTITUTE OF TECHNOLOGY

(An Autonomous Institution)

(Accredited by NAAC, NBA, Approved by AICTE New Delhi & Permanently Affiliated to JNTUH)

Aziznagar Gate, C.B. Post, Hyderabad-500 075'

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Continuous Lab Assessment sheet

Academic year: 2020-21 Regulation: R18

Course name: DS/PP Lab. Course code: A28508

Year/sem: II-I Section: C Date: 16/12/20

S. No	Ht.No	Observation (5M)	Completion of Experiment (10M)	Viva (5M)	Record (5M)	Total (25M)
1.	19911A05C1	—	—	A	—	—
2.	5C2	—	—	A	—	—
3.	5C3	5	10	4	5	24
4.	5C4	—	—	A	—	—
5.	5C5	5	10	3	5	23
6.	5C6	5	10	2	5	22
7.	5C7	—	—	A	—	—
8.	5C8	5	10	3	5	23
9.	5C9	5	10	1	5	21
10.	5D0	5	10	3	5	23
11.	5D1	5	10	4	5	24
12.	5D2	5	10	3	5	23
13.	5D3	5	10	2	5	22
14.	5D4	5	10	1	5	21
15.	5D5	—	—	A	—	—
16.	5D6	5	10	3	5	23
17.	5D7	5	10	4	5	24
18.	5D8	—	—	A	—	—
19.	5D9	5	10	5	5	25
20.	5E0	5	10	5	5	25
21.	5E1	5	10	4	5	24
22.	5E2	5	10	3	5	23
23.	5E3	5	10	2	5	22
24.	5E4	5	10	5	5	25
25.	5E5	5	10	4	5	24
26.	5E6	—	—	A	—	—
27.	5E7	5	10	2	5	22
28.	5E8	5	10	1	5	21
29.	5E9	—	—	A	—	—

30.	5F0	5	10	4	5	24
31.	5F1	5	10	3	5	23
32.	5F2	5	10	4	5	24
33.	5F3	—	—	A	—	—
34.	5F4	5	10	3	5	23
35.	5F5	—	—	A	—	—
36.	5F6	—	—	A	—	—
37.	5F7	—	—	A	—	—
38.	5F8	—	—	A	—	—
39.	5F9	—	—	A	—	—
40.	5G0	5	10	2	5	22
41.	5G1	5	10	1	5	21
42.	5G2	—	—	A	—	—
43.	5G3	5	10	5	5	25
44.	5G4	5	10	4	5	24
45.	5G5	5	10	3	5	23
46.	5G6	5	10	2	5	22
47.	5G7	—	—	A	—	—
48.	5G8	—	—	A	—	—
49.	5G9	5	10	1	5	21
50.	5H0	5	10	5	5	25
51.	5H1	5	10	5	5	25
52.	5H2	5	10	4	5	24
53.	5H3	5	10	3	5	23
54.	5H4	—	—	A	—	—
55.	5H5	5	10	3	5	23
56.	5H6	5	10	4	5	24
57.	5H7	5	10	3	5	23
58.	5H8	5	10	4	5	24
59.	5H9	—	—	A	—	—
60.	5J0	5	10	1	5	21
61.	18911A0558	—	—	A	—	—
62.	5B5	—	—	A	—	—
63.						
64.						
65.						

Note: At the end of the semester total marks are calculated to 15.

(Course Instructor)

VIDYA JYOTHI INSTITUTE OF TECHNOLOGY

(An Autonomous Institution)

(Accredited by NAAC, NBA, Approved by AICTE New Delhi & Permanently Affiliated to JNTUH)

Aziznagar Gate, C.B. Post, Hyderabad-500 075

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Continuous Lab Assessment sheet

Academic year: 2020-21 Regulation: R18

Course name: DS/PP Lab Course code: A23506

Year/sem: II-I Section: C Date: 12/2/2021

S. No	Ht.No	Observation (5M)	Completion of Experiment (10M)	Viva (5M)	Record (5M)	Total (25M)
1.	19911A05C1	—	—	AB	—	—
2.	5C2	—	—	AB	—	—
3.	5C3	5	10	4	5	24
4.	5C4	—	—	AB	—	—
5.	5C5	5	10	4	5	24
6.	5C6	5	10	3	5	23
7.	5C7	—	—	AB	—	—
8.	5C8	5	10	3	5	23
9.	5C9	5	10	2	5	22
10.	5D0	5	10	1	5	21
11.	5D1	5	10	4	5	24
12.	5D2	5	10	5	5	25
13.	5D3	5	10	2	5	22
14.	5D4	5	10	1	5	21
15.	5D5	—	—	AB	—	—
16.	5D6	5	10	5	5	25
17.	5D7	5	10	2	5	22
18.	5D8	—	—	AB	—	—
19.	5D9	—	—	AB	—	—
20.	5E0	5	10	3	5	23
21.	5E1	5	10	4	5	24
22.	5E2	5	10	1	5	21
23.	5E3	—	—	AB	—	—
24.	5E4	—	—	AB	—	—
25.	5E5	—	—	AB	—	—
26.	5E6	5	10	2	5	22
27.	5E7	5	10	5	5	25
28.	5E8	5	10	4	5	24
29.	5E9	—	—	AB	—	—

30.	19911405560	5	10	4	5	24
31.	5F1	5	10	3	5	23
32.	5F2	5	10	4	5	24
33.	5F3	—	—	AB	—	
34.	5F4	5	10	1	5	21
35.	5F5	5	10	2	5	22
36.	5F6	5	10	4	5	24
37.	5F7	—	—	AB	—	
38.	5F8	5	10	3	5	23
39.	5F9	5	10	2	5	22
40.	5G0	5	10	5	5	25
41.	5G1	5	10	4	5	24
42.	5G2	—	—	AB	—	
43.	5G3	5	10	2	5	22
44.	5G4	5	10	5	5	25
45.	5G5	5	10	4	5	24
46.	5G6	5	10	3	5	23
47.	5G7	—	—	AB	—	
48.	5G8	—	—	AB	—	
49.	5G9	5	10	1	5	21
50.	5H0	5	10	4	5	24
51.	5H1	5	10	3	5	23
52.	5H2	5	10	5	5	25
53.	5H3	5	10	2	5	22
54.	5H4	5	10	4	5	24
55.	5H5	5	10	2	5	22
56.	5H6	5	10	4	5	24
57.	5H7	5	10	5	5	25
58.	5H8	5	10	2	5	22
59.	5H9	5	10	3	5	23
60.	5I0	5	10	1	5	21
61.	19911405855	5	10	5	5	25
62.	556	—	—	AB	—	
63.						
64.						
65.						

Note: At the end of the semester total marks are calculated to 15.

(Course Instructor)

Vidya Jyothi Institute of Technology

(An Autonomous Institution)
(Accredited by NAAC & NBA, Approved by AICTE New Delhi & Permanently Affiliated to JNTUH)
Aziz Nagar Gate, C.B. Post, Hyderabad-500 075

Department of Computer Science & Engineering

Course End Survey Form Academic year: 2020-21

Name of the student	K Sai Vaishya	Year &sem	II– I
Roll number	1991A05E7	Regulations	R 19

Dear Student,

We need your help in evaluating the courses offered, by responding the short survey below.

Your feedback is very valuable for us in order to continually improve our program. The aim of this survey is to evaluate how well each of the courses has prepared you to have necessary skills.

Your responses will be kept confidential and will not be revealed to anyone outside the department without your permission.

Please indicate (√) the level to which you agree with the following criterion:
(3: Strongly agree 2: Agree 1: Strongly disagree)

Name of The Course: DS & PP Lab		RATING		
After completing this course the student must demonstrate the knowledge and ability to		3	2	1
CO 1	Develop the programs on stacks, trees and its applications	✓		
CO 2	Design and implement of programs on BST and Graph Traversals	✓		
CO 3	Apply hashing techniques in real world applications	✓		
CO 4	Implement oops concepts in Python	✓		
CO 5	Develop programs and modules and packages	✓		
CO 6	Design programs that handle errors	✓		

Any other comments / suggestions: _____

Signature

VIDYA JYOTHI INSTITUTE OF TECHNOLOGY

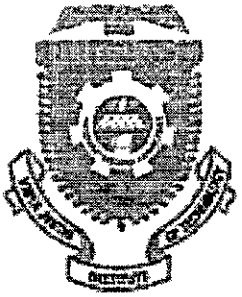
(Autonomous)

Aziz Nagar Gate, C.B. Post, Hyderabad-75

Department of Computer Science &Engineering

Year : II B.Tech **Semester : I**

DATA STRUCTURE LAB



LAB MANUAL

Head of the Department
Computer Science and Engineering
VJIT, Hyderabad-50075.

INDEX

Week 1:

Write a program to illustrate concepts of arrays, structures, unions and enumerated data types.

Arrays:

```
#include<stdio.h>

int main(){
int values[5];

  printf("Enter 5 integers: ");

// taking input and storing it in an array
for(int i =0; i<5;++i){
   scanf("%d",&values[i]);
}

  printf("Displaying integers: ");

// printing elements of an array
for(int i =0; i<5;++i){
    printf("%d\n", values[i]);
}
return0;
}
```

Output

```
Enter 5 integers: 1
-3   34    0    3
Displaying integers: 1
-3
34
0
3
```

Structures:

```
structPoint
{
  intx, y;
};

intmain()
{
  structPoint p1 = {0, 1};
   // Accessing members of point p1
```

```
  p1.x = 20;
  printf("x = %d, y = %d", p1.x, p1.y);

  return0;
}
```

Output:

x = 20, y = 1

Unions:

```
#include<stdio.h>
#include<string.h>

unionData{
int i;
float f;
char str[20];
};

int main(){

unionData data;

 printf("Memory size occupied by data : %d\n",sizeof(data));

return0;
}
```

Output:

Memory size occupied by data : 20

Enum:

```
#include<stdio.h>
enum week{Mon=10,Tue,Wed,Thur,Fri=10,Sat=16,Sun};
enum day{Mond,Tues,Wedn,Thurs,Frid=18,Satu=11,Sund};
int main(){
  printf("The value of enum week:
%d\t%d\t%d\t%d\t%d\t%d\t%d\n\n",Mon,Tue,Wed,Thur,Fri,Sat,Sun);
  printf("The default value of enum day:
%d\t%d\t%d\t%d\t%d\t%d\t%d",Mond,Tues,Wedn,Thurs,Frid,Satu,Sund);
return0;
}
```

Output

The value of enum week: 1011 12 13 10 16 17

The default value of enum day: 0 1 2 3 18 11 12

Week 2:

Aim: Write a C program that uses stack operations to convert a given infix expression into its postfix Equivalent, Implement the stack using an array.

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<conio.h>

char stack[20];
int top=-1;
char pop(); /*declaration of pop function*/
void push(char item); /*declaration of push function*/
intprcd(char symbol) /*checking the precedence*/
{
switch(symbol) /*assigning values for symbols*/
  {
case '+':
case '-': return 2;
                break;

        case '*':
        case '/':
        case '%': return 4;
                break;
        case '(':
        case ')':
        case '#': return 1;
                break;

 }
}
intisoperator(char symbol) /*assigning operators*/
{
switch(symbol)
  {
case '+':
```

```
case '*':
case '-':
case '/':
case '%':
case '(':
case ')':return 1;
        break;
default: return 0;
  }
}

/*converting infix to postfix*/
voidconvertip(char infix[],char postfix[])
{
        inti,n,j=0;
char symbol;
        top++;
        stack[top]='#';
        n=strlen(infix);
        for(i=0;i<strlen(infix);i++)
        {
        symbol=infix[i];
                if(isoperator(symbol)==0)
                {
                        postfix[j]=symbol;
                        j++;
                }
                else
                {
                        if(symbol=='(')
                        push(symbol); /*function call  for pushing elements into the stack*/
                        else if(symbol==')')
                        {
                                while(stack[top]!='(')
                                {
                                postfix[j]=pop();
                                j++;
                                }
```

```
                        pop(); /*function call  for popping elements into the stack*/
                    }
                    else
                    {
                    if(prcd(symbol)>prcd(stack[top]))
                    push(symbol);
                    else
                     {
                                while(prcd(symbol)<=prcd(stack[top]))
                                {
                                postfix[j]=pop();
                                j++;
                                }
                                push(symbol);

                     }/*end of else */
                    }/*end of else */
                } /*end of else */
        }/*end of for loop*/

        while(stack[top]!='#')
        {
                postfix[j]=pop();
                j++;
        }
        postfix[j]='\0'; /*null terminate string*/
}

/*main program*/
void main()
{

char infix[20],postfix[20];
//clrscr();
        printf("enter the valid infix string \n");
        gets(infix);
convertip(infix,postfix);   /*function call   for converting infix to postfix */
printf("the corresponding postfix string is:\n");
```

```
puts(postfix);
getch();
 }

/*push operation*/
void push(char item)
{
top++;
        stack[top]=item;
}
/*pop operation*/
char pop()
{
char a;
 a=stack[top];
top--;
return a;
 }
```

Output:

```
enter the valid infix string
9-((3*4)+8/4
the corresponding postfix string is: 934*8+-4/
```

Week 3:

Aim: Write a C program to evaluate postfix notations

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
#define MAX 50
int stack[MAX];
char postfix[20];
int top=-1;
void push(int item);
char pop();
void evaluate(char s);
void main()
{
inti,n,x;
char symbol;
// clrscr();
printf("Insert a postfix notation :: \n");
gets(postfix);
 n=strlen(postfix);
for(i=0;i<n;i++)
        {
symbol=postfix[i];
                if(symbol>='0' && symbol<='9')
                 {
                  x=(int)(symbol-48);
                push(x);
                }
                else
                 {
                evaluate(postfix[i]);
                 }
 }
printf("\n\nResult is :: %d",stack[top]);
getch();
}
```

```
void push(int x)
{
top++;
stack[top]=x;
}
char pop()
{
        char x;
         x=stack[top];
        top--;
        return x;
}

void evaluate(char d)
{
inta,b,c;
 a=pop();
 b=pop();

        switch(d)
        {
                case '+':
                         c=a+b;
                        top++;
                        stack[top]=c;
                        break;
                case '-':
                        c=a-b;
                        top++;
                        stack[top]=c;
                        break;
                case '*':
                        c=a*b;
                        top++;
                        stack[top]=c;
                        break;
                case '/':
                        c=a/b;
```

```
                top++;
                stack[top]=c;
                break;
        case '%':
                c=a%b;
                break;
                top++;
                stack[top]=c;
        defaul]t:
                 c=0;
            }

}
```

OUTPUT:

Insert a postfix notation: 432+*
Result is: 20

Week 4:

Aim: C program to illustrate tree traversals
a) In order b) preorder c) post order

In-order traversal method:
1. Visit Left Sub-Tree
2. Process Current Node
3. Visit Right Sub-Tree

Pre-order traversal method:
1. Process Current Node
2. Visit Left Sub-Tree
3. Visit Right Sub-Tree

Post-order traversal method:
1. Visit Left Sub-Tree
2. Visit Right Sub-Tree
3. Process Current Node

```
#include<stdio.h>
#include<stdlib.h>

struct node
{
int value;
node* left;
node* right;
};

struct node* root;

struct node* insert(struct node* r,int data);
voidinOrder(struct node* r);
voidpreOrder(struct node* r);
voidpostOrder(struct node* r);

int main()
{
root= NULL;
int n, v;

printf("How many data's do you want to insert ?\n");
```

```
scanf("%d",&n);

for(int i=0; i<n; i++){
printf("Data %d: ", i+1);
scanf("%d",&v);
root= insert(root, v);
}

printf("Inorder Traversal: ");
inOrder(root);
printf("\n");

printf("Preorder Traversal: ");
preOrder(root);
printf("\n");

printf("Postorder Traversal: ");
postOrder(root);
printf("\n");


return0;
}

struct node* insert(struct node* r,int data)
{
if(r==NULL)
{
     r =(struct node*)malloc(sizeof(struct node));
r->value = data;
r->left = NULL;
r->right = NULL;
}
elseif(data < r->value){
r->left = insert(r->left, data);
}
else{
r->right = insert(r->right, data);
}
return r;

}

voidinOrder(struct node* r)
```

```
{
if(r!=NULL){
inOrder(r->left);
printf("%d ", r->value);
inOrder(r->right);
}
}

voidpreOrder(struct node* r)
{
if(r!=NULL){
printf("%d ", r->value);
preOrder(r->left);
preOrder(r->right);
}
}

voidpostOrder(struct node* r)
{
if(r!=NULL){
postOrder(r->left);
postOrder(r->right);
printf("%d ", r->value);

}
}
```

OUTPUT:

```
How many data's do you want to insert? 7
20 15 25 16 24 12 30

Inorder Traversal:    12  15 16 20 24 25  30
Preorder Traversal:   20 15 12 16  25  24 30
Postorder Traversal:  12  16  15  24  30 25 20
```

Week 5:

Aim: C program to illustrate insert, delete and searching operation in binary search tree

```
#include<stdio.h>
#include<stdlib.h>

struct node
{
        int key;
        struct node *left, *right;
};

// A utility function to create a new BST node
struct node *newNode(int item)
{
        struct node *temp = (struct node *)malloc(sizeof(struct node));
        temp->key = item;
        temp->left = NULL;
        temp->right = NULL;
        return temp;
}

// A utility function to do inorder traversal of BST
voidinorder(struct node *root)
{
        if (root != NULL)
        {
                inorder(root->left);
                printf("%d ", root->key);
                inorder(root->right);
        }
}


struct node* insert(struct node* root, int key)
{
        /* If the tree is empty, return a new node */
```

```
        if (root == NULL)
                returnnewNode(key);

        if (key < root->key)
        root->left = insert(root->left, key);
        else
                root->right = insert(root->right, key);

        /* return the (unchanged) node pointer */
        return root;
}

struct node * Findmin(struct node* node)
{
        struct node* current = node;

        /* loop down to find the leftmost leaf */
        while (current->left != NULL)
                current = current->left;
        return current;
}

struct node* deletion(struct node* root, int key)
{
        // base case
        if (root == NULL) return root;

        // If the key to be deleted is smaller than the root's key,
        // then it lies in left subtree
        if (key < root->key)
                root->left = deletion(root->left, key);

        // If the key to be deleted is greater than the root's key,
        // then it lies in right subtree
        else if (key > root->key)
                root->right = deletion(root->right, key);
```

```
        else
        {
                // node with only one child or no child
                if (root->left == NULL)
                {
                        struct node *temp = root->right;
                        free(root);
                        return temp;
                }
                else if (root->right == NULL)
                {
                        struct node *temp = root->left;
                        free(root);
                        return temp;
                }

                // smallest element from thee right subtree
                struct node* temp = Findmin(root->right);

                // Copy the inorder successor's content to this node
                root->key = temp->key;

                // Delete the inorder successor
                root->right = deletion(root->right, temp->key);
        }
        return root;
}
// main program
void main()
{
        intdata,ch;
        struct node *root = NULL;
        while(1)
        {
                printf("\nenetr 1.insertion 2.display 3.deletion. 4 exit\n");
                printf("enter your choice\n");
                scanf("%d",&ch);
```

```
switch(ch)
{

        case 1: printf("\nenter the value to be inserted\n");
                scanf("%d",&data);
                root = insert(root, data);
                break;
        case 2: printf("\nInorder traversal of the given tree is: \n");
                inorder(root);
                break;

        case 3: printf("\nenter the value to be deleted\n");
                scanf("%d",&data);
                root = deletion(root, data);
                break;
        case 4: exit(0);

}
        }
}
```

OUTPUT:

```
enetr 1.insertion 2.display 3.deletion. 4 exit
enter your choice
1
enter the value to be inserted
20
enetr 1.insertion 2.display 3.deletion. 4 exit
enter your choice
1
enter the value to be inserted
25
enetr 1.insertion 2.display 3.deletion. 4 exit
enter your choice
1
enter the value to be inserted
15
enetr 1.insertion 2.display 3.deletion. 4 exit
enter your choice
```

1
enter the value to be inserted
12
enetr 1.insertion 2.display 3.deletion. 4 exit
enter your choice
1
enter the value to be inserted
16

enetr 1.insertion 2.display 3.deletion. 4 exit
enter your choice
2
enter the value to be deleted

15
enetr 1.insertion 2.display 3.deletion. 4 exit
enter your choice
2
Inorder traversal of the given tree is: 12 16 20 24 25 30

enetr 1.insertion 2.display 3.deletion. 4 exit
enter your choice
4

Week 6:

Aim: C Program to illustrate insertion, deletion and rotations on AVL Trees

```
#include<stdio.h>
#include<stdlib.h>

// An AVL tree node
struct node
{
        int key;
        struct node *left;
        struct node *right;
        int height;
};

// A utility function to get maximum of two integers
int max(int a, int b);

// A utility function to get height of the tree
int height(struct node *N)
{
        if (N == NULL)
                return 0;
        return N->height;
}

// A utility function to get maximum of two integers
int max(int a, int b)
{
        return (a > b)? a : b;
}

/* 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
// See the diagram given above.
struct node *rightRotate(struct node *y)
{
        struct node *x = y->left;
        struct node *T2 = x->right;

        // Perform rotation
        x->right = y;
        y->left = T2;

        // 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 x
// See the diagram given above.
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;
}

// Get Balance factor of node N
intgetBalance(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 the balance factor of this ancestor node to check whether
    this node became unbalanced */
    int balance = getBalance(node);

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

    // Left Left Case
    if (balance > 1 && key < node->left->key)
```

```
            returnrightRotate(node);
        // Right Right Case
        if (balance < -1 && key > node->right->key)
            returnleftRotate(node);

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

        // Right Left Case
        if (balance < -1 && key < node->right->key)
        {
            node->right = rightRotate(node->right);
            returnleftRotate(node);
        }

        /* return the (unchanged) node pointer */
        return node;
}

// A utility function to print preorder traversal of the tree.
// The function also prints height of every node
voidpreOrder(struct node *root)
{
        if(root != NULL)
        {
            printf("%d ", root->key);
            preOrder(root->left);
            preOrder(root->right);
        }
}

/* Drier program to test above function*/
void main()
{
```

```
        intdata,ch;
        struct node *root = NULL;
        while(1)
        {
                printf("\nenetr 1.insertion 2.display 3.deletion. 4 exit\n");
                printf("enter your choice\n");
                scanf("%d",&ch);
                switch(ch)
                {

                        case 1: printf("\nenter the value to be inserted\n");
                                scanf("%d",&data);
                                root = insert(root, data);
                                break;
                        case 2: printf("\n preorder traversal of the given tree is: \n");
                                preOrder(root);
                                break;

                        /*case 3: printf("\nenter the value to be deleted\n");
                                scanf("%d",&data);
                                root = deletion(root, data);
                                break;*/
                        case 4: exit(0);


                }
        }
}
```

OUTPUT:

```
enetr 1.insertion 2.display 3.deletion. 4 exit
enter your choice
1
20

enetr 1.insertion 2.display 3.deletion. 4 exit
enter your choice
```

1
15

enetr 1.insertion 2.display 3.deletion. 4 exit
enter your choice
1
12

enetr 1.insertion 2.display 3.deletion. 4 exit
enter your choice
2
preorder traversal of the given tree is:

15 12 20

enetr 1.insertion 2.display 3.deletion. 4 exit
4

Week 7:

Aim:Write a 'C' program to implement Breadth first search.

```
#include <stdio.h>
#include<conio.h>
voidcreategraph();
voidbfs(); void display();
int g[10][10],n;
void main()
{
        int v;
        clrscr();
        creategraph();
        printf("starting vertex is");
        scanf("%d",&v);
        bfs(v);
        getch();
}
voidcreategraph()
{
        inti,j;
        printf("enter the number of nodes");
        scanf("%d",&n);
        for(i=0;i",v);
        while(f<=r)
        {
                v=q[f];
                f++;
                for(i=0;i",i);
                visited[i]=1;
                q[++r]=i;
        }
}
```

Output:

```
Enter no of nodes 3
Edge present between A&A  0
Edge present between A&B  1
Edge present between A&C  1
Edge present between B&A  1
Edge present between B&B  0
```

Aim:Write a 'C' program to implement Depth first search.

```
#include<stdio.h>
#include <conio.h>
voidcreategraph();
voiddfs();
void display();
int g[10][10],n;
void main()
{
        int v;
        clrscr();
        creategraph();
        printf("starting vertex is");
        scanf("%d",&v);
        dfs(v);
        getch();
}
voidcreategraph()
{
        inti,j;
        printf("enter the number of nodes");
        scanf("%d",&n);
        for(i=0;i=0)
        {
                v=st[top];
                top--;
                if(visited [v]==0)
                {
                        printf("%d->",v);
                        visited [v]=1;
                }
                for(i=n-1;i>=0;i--)
                {
                        if(g[v][i]!=0 && visited[i]==0)
                        {
                                st[++top]=i;
                        }
                }}}
```

Output:

```
Enter no.of nodes 3
Edge present b/w A&A  0
```

```
Edge present b/w A&B  1
Edge present b/w A&C  1
Edge present b/w A&D  0
Edge present b/w B&A  1
Edge present b/w B&B  0
Edge present b/w B&C  0
Edge present b/w B&D  1
Edge present b/w C&A  1
Edge present b/w C&B  0
Edge present b/w C&C  0
Edge present b/w C&D  0
Edge present b/w D&A  0
Edge present b/w D&B  1
Edge present b/w D&C  0
Edge present b/w D&D  0
Enter vertex from above graph
0
A->B->D->C->
```

	A	B	C	D
A	0	1	1	0
B	1	0	0	1
C	1	0	0	0
D	0	1	0	0

Week 8:

Write a program to implement hash table using linear and quadratic probing

```
#include <stdio.h>
#include<stdlib.h>
#define TABLE_SIZE 10

int h[TABLE_SIZE]={NULL};

void insert()
{

 int key,index,i,flag=0,hkey;
 printf("\nenter a value to insert into hash table\n");
 scanf("%d",&key);
 hkey=key%TABLE_SIZE;
 for(i=0;i<TABLE_SIZE;i++)
   {

    index=(hkey+i)%TABLE_SIZE;

    if(h[index] == NULL)
    {
      h[index]=key;
       break;
    }

   }

   if(i == TABLE_SIZE)

    printf("\nelement cannot be inserted\n");
}
void search()
{

 int key,index,i,flag=0,hkey;
 printf("\nenter search element\n");
```

```
 scanf("%d",&key);
 hkey=key%TABLE_SIZE;
 for(i=0;i<TABLE_SIZE; i++)
 {
   index=(hkey+i)%TABLE_SIZE;
   if(h[index]==key)
   {
    printf("value is found at index %d",index);
    break;
   }
 }
 if(i == TABLE_SIZE)
   printf("\n value is not found\n");
}
void display()
{

 int i;

 printf("\nelements in the hash table are \n");

 for(i=0;i< TABLE_SIZE; i++)

 printf("\nat index %d \t value =  %d",i,h[i]);

}
main()
{
   int opt,i;
   while(1)
   {
      printf("\nPress 1. Insert\t 2. Display \t3. Search \t4.Exit \n");
      scanf("%d",&opt);
      switch(opt)
      {
         case 1:
            insert();
            break;
```

```
        case 2:
           display();
           break;
        case 3:
           search();
           break;
        case 4:exit(0);
      }
    }
}
```

Output

```
Press 1. Insert   2. Display   3. Search   4.Exit
1

enter a value to insert into hash table
12

Press 1. Insert   2. Display   3. Search   4.Exit
1

enter a value to insert into hash table
13

Press 1. Insert   2. Display   3. Search   4.Exit
1

enter a value to insert into hash table
22

Press 1. Insert   2. Display   3. Search   4.Exit
2

elements in the hash table are

at index 0    value =  0
at index 1    value =  0
```

```
at index 2    value =  12
at index 3    value =  13
at index 4    value =  22
at index 5    value =  0
at index 6    value =  0
at index 7    value =  0
at index 8    value =  0
at index 9    value =  0
Press 1. Insert    2. Display    3. Search    4.Exit
3

enter search element
12
value is found at index 2
Press 1. Insert    2. Display    3. Search    4.Exit
2 3

enter search element
23

value is not found

Press 1. Insert    2. Display    3. Search    4.Exit
```

Hashing using quadratic probing:

```
#include<stdio.h>
#include<stdlib.h>

/* to store a data (consisting of key and value) in hash table array */
struct item
{
   int key;
   int value;
};

/* each hash table item has a flag (status) and data (consisting of key and value) */
struct hashtable_item
{

   int flag;
   /*
    * flag = 0 : data does not exist
    * flag = 1 : data exists at given array location
    * flag = 2 : data was present at least once
    */
   struct item *data;

};

struct hashtable_item *array;
int size = 0;
int max = 10;

/* this function returns corresponding index of the given key */
int hashcode(int key)
{
   return (key % max);
}

/* this  function initializes the hash table array */
void init_array()
{
   int i;
   for (i = 0; i < max; i++)
     {
              array[i].flag = 0;
```

```
            array[i].data = NULL;
    }
}

/* this function inserts an element in the hash table */
void insert(int key, int value)
{
    int index = hashcode(key);
    int i = index;
    int h = 1;
    struct item *new_item = (struct item*) malloc(sizeof(struct item));
    new_item->key = key;
    new_item->value = value;

    /* probing through the array until an empty space is found */
    while (array[i].flag == 1)
      {
            if (array[i].data->key == key)
          {
                  /* case when already present key matches the given key */
                  printf("\n This key is already present in hash table, hence updating i
t's value \n");
                  array[i].data->value = value;
                  return;

            }
            i = (i + (h * h)) % max;
            h++;
            if (i == index)
       {
            printf("\n Hash table is full, cannot add more elements \n");
    return;
            }
    }
    array[i].flag = 1;
    array[i].data = new_item;
    printf("\n Key (%d) has been inserted\n", key);
    size++;
}
/* to remove an element form the hash table array */
void remove_element(int key)
{
    it index = hashcode(key);
```

```
    int i = index;
    int h = 1;
    /* probing through the hash table until we reach at location where there had not been
an element even once */
    while (array[i].flag != 0)
      {
            if (array[i].flag == 1  &&  array[i].data->key == key)
          {

                  /* case where data exists at the location and its key matches to the
given key */
                  array[i].flag = 2;
                  array[i].data = NULL;
                  size--;
                  printf("\n Key (%d) has been removed \n", key);
                  return;

            }
            i = (i + (h * h)) % max;
            h++;
            if (i == index)
 {
                  break;
            }
    }
    printf("\n Key does not exist \n");
}
/* to display the contents of hash table */
void display()
{
    int i;
    for(i = 0; i < max; i++)
      {
            if (array[i].flag != 1)
          {
                  printf("\n Array[%d] has no elements \n", i);
            }
            else
          {
                  printf("\n Array[%d] has elements \n  %d (key) and %d (value) \n", i
, array[i].data->key, array[i].data->value);
            }
    }}
```

```
int size_of_hashtable()
{
   return size;
}

void main()
{
   int choice, key, value, n, c;
   clrscr();
 array = (struct hashtable_item*) malloc(max * sizeof(struct hashtable_item*));
   init_array();
   do {
            .printf("Implementation of Hash Table in C with Quadratic Probing.\n\n");
             printf("MENU-: \n1.Inserting item in the Hash table"
                  "\n2.Removing item from the Hash table"
                  "\n3.Check the size of Hash table"
                  "\n4.Display Hash table"
                 "\n\n Please enter your choice-:");
            scanf("%d", &choice);
            switch(choice)
          {
            case 1:
                printf("Inserting element in Hash table \n");
                printf("Enter key and value-:\t");
                scanf("%d %d", &key, &value);
                insert(key, value);
            break;
            case 2:
                printf("Deleting in Hash table \n Enter the key to delete-:");
                scanf("%d", &key);
                remove_element(key);
            break;
            case 3:
                n = size_of_hashtable();
                printf("Size of Hash table is-:%d\n", n);
            break;
            case 4:
                display();
            break;
            default:
                 printf("Wrong.Input\n");
            }printf("\n Do you want to continue-:(press 1 for yes)\t");
            scanf("%d", &c);
```

```
        }while(c == 1);
        getch();
    }
```

Output:

```
Implementation of Hash Table in C with Quadratic Probing
MENU-:
1. Inserting item in the Hash table
2. Removing item from the Hash table
3. Check the size of Hash table
4. Display Hash table

Please enter your choice-: 3
Size of hash table is-: 0

Do you want to continue-:(press 1 for yes) 1
Implementation of Hash Table in C with Quadratic Probing
MENU-:
1. Inserting item in the Hash table
2. Removing item from the Hash table
3. Check the size of Hash table
4. Display Hash table

Please enter your choice-: 1
Inserting element in Hash table
Enter key and value-:  12   10

Key (12) has been inserted

Do you want to continue-:(press 1 for yes) 1
Implementation of Hash Table in C with Quadratic Probing
MENU-:
1. Inserting item in the Hash table
2. Removing item from the Hash table
3. Check the size of Hash table
4. Display Hash table

Please enter your choice-: 1
Inserting element in hash table
Enter key and value-:  122  4

Key (122) has been inserted

Do you want to continue-:(press 1 for yes) 1
```

```
Array[4] has no elements

Array[5] has no elements

Array[6] has no elements

Array[7] has no elements
82(key) and 5(value)
Array[8] has no elements

Array[9] has no elements


Do you want to continue-:(press 1 for yes) 1
Implementation of Hash Table in C with Quadratic Probing
MENU-:
1. Inserting item in the Hash table
2. Removing item from the Hash table
3. Check the size of Hash table
4. Display Hash table

Please enter your choice-: 2
Deleting in hash table
Enter the key to delete-: 122

 Key (122) has been removed

Do you want to continue-:(press 1 for yes) 1
Implementation of Hash Table in C with Quadratic Probing
MENU-:
1. Inserting item in the Hash table
2. Removing item from the Hash table
3. Check the size of Hash table
4. Display Hash table

Please enter your choice-: 2
Deleting in hash table
Enter the key to delete-: 56

 This key does not exist

Do you want to continue-:(press 1 for yes) 2
```

Implementation of Hash Table in C with Quadratic Probing
MENU-:
1. Inserting item in the Hash table
2. Removing item from the Hash table
3. Check the size of Hash table
4. Display Hash table

Please enter your choice-: 1
Inserting element in hash table
Enter key and value-: 82 5

Key (82) has been inserted

Do you want to continue-:(press 1 for yes) 1
Implementation of Hash Table in C with Quadratic Probing
MENU-:
1. Inserting item in the Hash table
2. Removing item from the Hash table
3. Check the size of Hash table
4. Display Hash table

Please enter your choice-: 3
Size of hash table is-: 3

Do you want to continue-:(press 1 for yes) 1
Implementation of Hash Table in C with Quadratic Probing
MENU-:
1. Inserting item in the Hash table
2. Removing item from the Hash table
3. Check the size of Hash table
4. Display Hash table

Please enter your choice-: 4
Array[0] has no elements

Array[1] has no elements

Array[2] has elements-:
12 (key) and 10 (value)

Array[3] has elements-:
122(key) and 4(value)

VIDYA JYOTHI INSTITUTE OF TECHNOLOGY

(Autonomous)

Aziz Nagar Gate, C.B. Post, Hyderabad-75

Department of Computer science & Engineering

Year : II B.Tech **Semester : I**

Python Programming Lab

(R20)

LAB MANUAL

Head of the Department
Computer Science and Engineering
Vidya Jyothi Institute of Technology

INDEX

Head of the Department
Computer Science and Engineering
VJIT, Hyderabad-50075.

1 a) Installation and Environment setup of python

Python distribution is available for a wide variety of platforms. You need to download only the binary code applicable for your platform and install Python.

steps to install Python

Open a Web browser and go to https://www.python.org/downloads

Follow the link for the Windows installer python-XYZ.msi file where XYZ is the version you need to install.

To use this installer python-XYZ.msi, the Windows system must support Microsoft Installer 2.0. Save the installer file to your local machine and then run it to find out if your machine supports MSI.

Run the downloaded file. This brings up the Python install wizard, which is really easy to use. Just accept the default settings, wait until the install is finished, and you are done

Setting up PATH

To add the Python directory to the path for a particular session

At the command prompt – type path %path%;C:\Python and press Enter.

Running Python

There are three different ways to start Python

- Interactive Interpreter
- Script from the Command-line
- Integrated Development Environment(PythonWin)

1 b) Write a program to demonstrate the use of basic Data Types

```
a = 5
print(a, "is of type", type(a))
a = 2.0
print(a, "is of type", type(a))
a = 1+2j
print(a, "is complex number?", isinstance(1+2j,complex))
```

Output:

```
5 is of type <class 'int'>
2.0 is of type <class 'float'>
(1+2j) is complex number? True
```

1 c) Write a program to demonstrate the Operators and Expressions

Expressions:

```
# Solve the quadratic equation ax**2 + bx + c = 0

# import complex math module
importcmath
 To take coefficient input from the users
 a = float(input('Enter a: '))
 b = float(input('Enter b: '))
 c = float(input('Enter c: '))
# calculate the discriminant
d = (b**2) - (4*a*c)
# find two solutions
sol1 = (-b-cmath.sqrt(d))/(2*a)
sol2 = (-b+cmath.sqrt(d))/(2*a)

print('The solution are {0} and {1}'.format(sol1,sol2))
```

output:

```
 Enter a: 1
Enter b: 5
Enter c: 6
The solutions are (-3+0j) and (-2+0j)
```

1 d) Write a program to demonstrate the Functions and parameter passing Techniques.

```
# Call by value in Python
#defining the function
def sum (a,b):
returna+b;


#taking values from the user
a = int(input("Enter a: "))
b = int(input("Enter b: "))
 #printing the sum of a and b
print("Sum = ",sum(a,b))
```

Output:

```
Enter a: 10
Enter b: 20
Sum = Enter a: 10
Enter b: 20
Sum =  30
```

Call by reference in Python

```
#defining the function
defchange_list(list1):
list1.append(20);
list1.append(30);
print("list inside function = ",list1)

#defining the list
list1 = [10,30,40,50]

#calling the function
change_list(list1);
print("list outside function = ",list1);
```

Output:

list inside function = [10, 30, 40, 50, 20, 30]

list outside function = [10, 30, 40, 50, 20, 30]

Exercise 2

a)Write a Program to implement

i. Packages ii. Modules iii. Built-in Functions

I) Packages:

ITEmployees.py

```
defgetITNames():
   List = ["Rajashekar", "Narayana", "Tharak",   "Prabhu"]
return List;
```

__init__.py

```
fromITEmployees import getITNames
```

Test.py

```
import Employees
print(Employees.getNames())
```

Output:

```
['Rajashekar', 'Narayana', 'Tharak', 'Prabhu']
```

II)Python Modules:

calculation.py:

```
def summation(a,b):
returna+b;
def multiplication(a,b):
return a*b ;
def divide(a,b):
return a/b ;
```

Main.py

```
from calculation import summation
#it will import only the summation() from calculation.py
a = int(input("Enter the first number"))
b = int(input("Enter the second number"))
print("Sum = ",summation(a,b))
```

Output:

```
Enter the first number 10
Enter the second number 20
Sum =  30
```

III) Built-in Functions

```
numbers = [2.5, 3, 4, -5]
# start parameter is not provided
numbersSum = sum(numbers)
print(numbersSum)
# start = 10
numbersSum = sum(numbers, 10)
print(numbersSum)
integer = -20
print('Absolute value of -20 is:', abs(integer))
floating = -30.33
print('Absolute value of -30.33 is:', abs(floating))
```

output:

```
4.5
14.5
Absolute value of -20 is: 20
Absolute value of -30.33 is: 30.33
```

b) Write a Program to implement

i. List ii. Tuple iii. Dictionaries

i. List

```
l =[];

n = int(input("Enter the number of elements in the list")); #Number of elements will be entered
#by the user

fori in range(0,n): # for loop to take the input

l.append(input("Enter the item?")); # The input is taken from the user and added to the list as
#the item

print("printing the list items....");

fori in l: # traversal loop to print the list items

print(i, end = " ");
```

```
Enter the number of elements in the list 5

Enter the item?1

Enter the item?2

Enter the item?3

Enter the item?4

Enter the item?5

printing the list items....

1 2 3 4 5
```

ii. Tuple

```
tuple1 = tuple(input("Enter the tuple elements ..."))
print(tuple1)
count = 0
fori in tuple1:
print("tuple1[%d] = %s"%(count, i));
```

Output:

```
Enter the tuple elements ...12345
('1', '2', '3', '4', '5')
tuple1[0] = 1
tuple1[0] = 2
tuple1[0] = 3
tuple1[0] = 4
tuple1[0] = 5
```

OR

```
# empty tuple
my_tuple = ()
print(my_tuple)

# tuple having integers
my_tuple = (1, 2, 3)
print(my_tuple)

# tuple with mixed datatypes
my_tuple = (1, "Hello", 3.4)
print(my_tuple)
```

```
# nested tuple
my_tuple = ("mouse", [8, 4, 6], (1, 2, 3))
print(my_tuple)

# tuple can be created without parentheses
my_tuple = 3, 4.6, "dog"
print(my_tuple)

a, b, c = my_tuple
print(a)
print(b)
print(c)
```

Output:

```
()
(1, 2, 3)
(1, 'Hello', 3.4)
('mouse', [8, 4, 6], (1, 2, 3))
(3, 4.6, 'dog')
3
4.6
dog
```

iii. Dictionaries

```
Employee = {"Name": "John", "Age": 29, "salary":25000,"Company":"GOOGLE"}
print(type(Employee))
print("printing Employee data .... ")
for x in Employee.items():
print(x)

print("Enter the details of the new employee....");
Employee["Name"] = input("Name: ");
Employee["Age"] = int(input("Age: "));
Employee["salary"] = int(input("Salary: "));
Employee["Company"] = input("Company:");
print("printing the new data");
forx,y in Employee.items():
print(x,y)
```

output:

```
<class 'dict'>
printing Employee data ....
('Name', 'John')
('Age', 29)
('salary', 25000)
('Company', 'GOOGLE')
Enter the details of the new employee....
Name: David
Age: 19
Salary: 8900
Company:Wipro
```

printing the new data

Name David

Age 19

Salary 8900

Company Wipro

2 c) Programs on Stings, String Operations and Regular Expressions

i) #Counting a particular letter in a given string

```
count = 0

for letter in 'Hello World':

    if(letter == 'l'):

    count += 1

print(count,'letters found')
```

output:

3 letters found

ii) #sorting words

```
my_str = input("Enter a string: ")

words = my_str.split()

words.sort()

print("The sorted words are:")

for word in words:

print(word)
```

output:

Enter a string

Welcome to python programming

The sorted words are: Programming

Python

To

Welcome

Program to check if a string is palindrome or not

```
my_str = 'aIbohPhoBiA'
my_str = my_str.casefold()
rev_str = reversed(my_str)
if list(my_str) == list(rev_str):
print("It is palindrome")
else:
print("It is not palindrome")
```

Output:

It is palindrome

iii)Regular Expressions

Program to remove all whitespaces

```
import re
# multiline string
string = 'abc 12\
de 23 \n f45 6'
# matches all whitespace characters
pattern = '\s+'
# empty string
replace = ''
new_string = re.subn(pattern, replace, string)
print(new_string)
```

Output:

('abc12de23f456', 4)

3 a) Write a Program to implement Class and Object

```
classMyClass:
        "This is my second class"
        a = 10
        deffunc(self):
                print('Hello')


ob = MyClass()
print(MyClass.func)
print(ob.func)
ob.func()
```

Output

```
<functionMyClass.func at 0x7fc988389bf8>
<bound method MyClass.func of <__main__.MyClass object at 0x7fc9883a40f0>>
Hello
```

3 b) Write a Program to implement Static and Instance methods,Abstract Classes and Interfaces

I) **#Static and Instance methods**

```
fromdatetime import date
class Person:
def __init__(self, name, age):
    self.name = name
self.age = age

  # a class method to create a Person object by birth year.
  @classmethod
deffromBirthYear(cls, name, year):
returncls(name, date.today().year - year)

  # a static method to check if a Person is adult or not.
  @staticmethod
defisAdult(age):
return age > 18

person1 = Person('mayank', 21)
person2 = Person.fromBirthYear('mayank', 1996)

print person1.age
print person2.age
printPerson.isAdult(22)
21
21
True
```

 # Abstract Classes and Interfaces

```
fromabc import ABC, abstractmethod
class Polygon(ABC):
    # abstract method
defnoofsides(self):
pass
class Triangle(Polygon):

  # overriding abstract method
defnoofsides(self):
print("I have 3 sides")

class Pentagon(Polygon):

  # overriding abstract method
defnoofsides(self):
print("I have 5 sides")

class Hexagon(Polygon):

  # overriding abstract method
defnoofsides(self):
print("I have 6 sides")
```

```
class Quadrilateral(Polygon):


   # overriding abstract method
defnoofsides(self):
print("I have 4 sides")


# Driver code
R = Triangle()
R.noofsides()


K = Quadrilateral()
K.noofsides()


R = Pentagon()
R.noofsides()


K = Hexagon()
K.noofsides()
```

Output:

```
I have 3 sides
I have 4 sides
I have 5 sides
I have 6 sides
```

4 a) Write a program to compute distance between two points taking input from the user (Pythagorean Theorem)

```
import math
x1 = int(input(" co_ordinates X1 => "))
y1 = int(input(" co_ordinates Y1 => "))
x2 = int(input(" co_ordinates X2 => "))
y2 = int(input(" co_ordinates Y2 => "))
x = (x1 - x2) ** 2
y = (y1 - y2) ** 2
distance = math.sqrt(x + y)
print('Result is ')
print(distance)
```

Output:

```
co_ordinates X1 => 2
co_ordinates Y1 => 5
co_ordinates X2 => 5
co_ordinates Y2 => 9
Result is 5.0
```

4 b) Write a program to convert a given decimal number to other base systems

```
dec = int(input("Enter a number "));


print("The decimal value of",dec,"is:")
print(bin(dec),"in binary.")
print(oct(dec),"in octal.")
print(hex(dec),"in hexadecimal.")
```

Output:

```
Enter a number
344
The decimal value of 344 is:
0b101011000 in binary.
0o530 in octal.
0x158 in hexadecimal.
```

5 a) Write a program to implement Inheritance

```
class Polygon:
def __init__(self, no_of_sides):
self.n = no_of_sides
self.sides = [0 for i in range(no_of_sides)]

definputSides(self):
self.sides = [float(input("Enter side "+str(i+1)+" : ")) for i in range(self.n)]

defdispSides(self):
fori in range(self.n):
print("Side",i+1,"is",self.sides[i])


class Triangle(Polygon):
def __init__(self):
    Polygon.__init__(self,3)

deffindArea(self):
a, b, c = self.sides
    # calculate the semi-perimeter
    s = (a + b + c) / 2
area = (s*(s-a)*(s-b)*(s-c)) ** 0.5
print('The area of the triangle is %0.2f' %area)

t = Triangle()

t.inputSides()
t.dispSides()
t.findArea()
```

output:

```
Enter side 1 : 3
Enter side 2 : 5
Enter side 3 : 4

Side 1 is 3.0
Side 2 is 5.0
Side 3 is 4.0
The area of the triangle is 6.00
```

5 b) Write a program to implement Polymorphism

```
class Document:
def __init__(self, name):
self.name = name

def show(self):
raiseNotImplementedError("Subclass must implement abstract method")

class Pdf(Document):
def show(self):
return 'Show pdf contents!'

class Word(Document):
def show(self):
return 'Show word contents!'

documents = [Pdf('Document1'),
Pdf('Document2'),
Word('Document3')]

for document in documents:
print document.name + ': ' + document.show()
```

Output:

Document1: Show pdf contents!

Document2: Show pdf contents!

Document3: Show word contents!

6 a) Write a program to implement Files

```
f = open("demofile.txt", "r")
print(f.read())
# Return the 5 first characters of the file:
print(f.read(5))
# Read one line of the file:
print(f.readline())
# Loop through the file line by line
for x in f:
print(x)

# Write the content to a file
f2 = open("demofile.txt", "a")
f2.write("Now the file has one more line!")
```

6 b) Write a program to Exception Handling.

```
try:
   a = int(input("Enter a:"))
   b = int(input("Enter b:"))
   c = a/b;
print("a/b = %d"%c)
except:
print("can't divide by zero")
else:
print("Hi I am else block")
```

output:

```
Enter a:10
Enter b:2
a/b = 5
Hi I am else block

Enter a:10
Enter b:0
can't divide by zero
```