Sri Indu
College of Engineering & Technology
UGC Autonomous Institution
Recognized under 2(f) & 12(B) of UGC Act 1956,
NAAC, Approved by AICTE &
Permanently Affiliated to JNTUH
Estd.2001

NAAC
NATIONAL ASSESSMENT AND ACCREDITATION COUNCIL

LAB MANUAL

DATA STRUCTURES

R20CSE21L1

II Year I Semester

DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING

ACADEMIC YEAR 2022-2023

SRI INDU COLLEGE OF ENGINEERING & TECHNOLOGY

(An Autonomous Institution under UGC, New Delhi)

(Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi and Accredited by NBA, NAAC)
Sheriguda Village, Ibrahimpatnam Mandal, Ranga Reddy Dist. – 501 510

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Contents

SRI INDU COLLEGE OF ENGINEERING & TECHNOLOGY

(An Autonomous Institution under UGC, New Delhi)

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

Sheriguda Village, Ibrahimpatnam Mandal, Ranga Reddy Dist. – 501 510

I

I

INSTITUTION VISION

To be a premier institution in engineering & technology and management with competence, values and social consciousness.

INSTITUTION MISSION

IM$_1$: Provide high quality academic programmes, training activities and research facilities.

IM$_2$: Promote continuous industry-institute interaction for employability, entrepreneurship, leadership and research aptitude among stakeholders.

IM$_3$: Contribute to the economic and technological development of the region, state and nation.

SRI INDU COLLEGE OF ENGINEERING & TECHNOLOGY

(An Autonomous Institution under UGC, New Delhi)

(Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi and Accredited by NBA, NAAC)
Sheriguda Village, Ibrahimpatnam Mandal, Ranga Reddy Dist. – 501 510

VISION OF THE DEPARTMENT

To be a technologically adaptive centre for computing by grooming the students as top notch professionals.

MISSION OF THE DEPARTMENT

DM_1: To offer quality education in computing.

DM_2: To provide an environment that enables overall development of all the stakeholders.

DM_3: To impart training on emerging technologies like Data Analytics, Artificial Intelligence and Internet Of Things.

DM_4: To encourage participation of stakeholders in research and development

Program Educational Objectives(PEO's)

PEO1	**Higher Studies:** Graduates with an ability to apply knowledge of Basic Sciences and programming skills in their career and higher education.
PE02	**Lifelong Learning:** Graduates with an ability to adopt new technologies for ever changing IT industry needs through Self-Study, Critical thinking and Problem-solving skills.
PEO3	**Professional Skills:** Graduates will be ready to work in projects related to complex problems involving multidisciplinary projects with effective analytical skills
PEO4	**Engineering citizenship:** Graduates with an ability to communicate well and exhibit social, technical and ethical responsibility in process or product.

Program Specific Outcomes(PSO's)

PSO1	**Software Development:** To apply the knowledge of Software Engineering, Data Communication, Web Technology and Operating Systems for building IOT and Cloud Computing applications.
PSO2	**Industrial Skills Ability:** Design, develop and test software systems for world-wide network of computers to provide solutions to real world problems.
PSO3	**Project Implementation:** Analyze and recommend the appropriate IT infrastructure required for the implementation of a project.

Program Outcomes(PO's)

PO1	**Engineering Knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2	**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.
PO3	**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.
PO4	**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.
PO5	**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.
PO6	**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.
PO7	**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.
PO8	**Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
PO9	**Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10	**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.
PO11	**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.
PO12	**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.

SRI INDU COLLEGE OF ENGINEERING & TECHNOLOGY
(An Autonomous Institution under UGC, New Delhi)
Recognized under 2(f) and 12(B) of UGC Act 1956
NBA Accredited, Approved by AICTE and Permanently affiliated to JNTUH
Sheriguda (V), Ibrahimpatnam, R.R.Dist, Hyderabad - 501 510
2001

D4

BR-20

LR.NO.SICET/AUTO/DAE/BR-20/ACADEMIC-CAL/320/2022 DATE: 05.09.2022

II B.TECH ACADEMIC CALENDAR
ACADEMIC YEAR : 2022-2023

Dr. G. SURESH,
Principal,
To,
All the HODs.
Sir,

Sub: SICET (Autonomous) - Academic & Evaluation - Academic Calendar for **II B.Tech - I & II Semester** for the academic year **2022-23** – Reg.

****

The approved Academic Calendar for **II B.Tech – I & II Semester** for the academic Year **2022-23** is given below:

ACADEMIC CALENDAR - II B.TECH - I & II SEMESTER
ADMITTED BATCH – (2021 – 2022) of BR-20 Regulation.

I SEMESTER

Commencement of I Sem class work	26.09.2022	
I Spell of Instructions (Including Dussehra Holidays).	26.09.2022 - 26.11.2022	9 Weeks
Dussehra Holidays.	03.10.2022 - 08.10.2022	1 Week
I Mid Examinations for II B.Tech I Sem Students.	28.11.2022 - 30.11.2022	3 Days
II Spell of Instructions.	01.12.2022 - 28.01.2023	8 Weeks 3 Days
Sankranti Holidays.	13.01.2023 - 16.01.2023	4 Days
II Mid Examinations for II B.Tech I Sem Students.	30.01.2023 - 01.02.2023	3 Days
Preparation Holidays, Practical Lab Examinations and Remedial Mid Test (RMT).	02.02.2023 - 11.02.2023	10 Days
II B.Tech I Semester End Examinations (Main) and Supplementary Examinations.	13.02.2023 - 25.02.2023	2 Weeks
Commencement of Class-Work for II B.Tech - II Semester 27.02.2023 (Monday).		

II SEMESTER

Commencement of II Sem class work.	27.02.2023	
I Spell of Instructions.	27.02.2023 - 22.04.2023	8 Weeks
I Mid Examinations for II B.Tech. II Sem. Students.	24.04.2023 - 26.04.2023	3 Days
II Spell of Instructions.	27.04.2023 - 05.07.2023	10 Weeks
Summer Vacation.	15.05.2023 - 27.05.2023	2 Weeks
II Mid Examinations for II B.Tech. II Sem. Students.	06.07.2023 - 08.07.2023	3 Days
Preparation Holidays, Practical Lab Examinations Remedial Mid Test (RMT).	10.07.2023 - 19.07.2023	10 Days
II B.Tech II Semester End Examinations (Main) and Supplementary Examinations.	20.07.2023 - 02.08.2023	2 Weeks
Commencement of Class-Work for III B.Tech - I Semester 07.08.2023 (Monday).		

ACE CE DIRECTOR PRINCIPAL

Copy to: All the Heads of the Depts., A.O. Sir.

CONTROLLER OF EXAMINATIONS
Sri Indu College of Engineering & Technology
(An Autonomous Institution under JNTUH)
Sheriguda (V), Ibrahimpatnam, R.R. Dist-501510.

DIRECTOR
(Academic Audit)
Sri Indu College of Engineering & Technology
Sheriguda, IBP, R.R. Dist-501510.

PRINCIPAL
Sri Indu College of Engineering & Technology
(An Autonomous Institution Under JNTUH)
Sheriguda (V), Ibrahimpatnam, R.R. Dist-501510

COURSEOUTCOMES(CO's)

AcademicYear:2021-22

Class:II YEAR-ISEM**.**

Course Name: Data Structures Lab (R20CSE21L1)

At the end ofthecourse, the studentwillbeable to

CourseOutcomes (COs)	
C21L1.1	Design a program to implement the linear data structures using static and dynamic memory allocation. (Create)**)**
C21L1.2	Design a program to implement searching ,sorting techniques for the given problem.(Create)
C21L1.3	Demonstrate the fundamental algorithms of tree data structures by experimenting the programs.(Apply)
C21L1.4	Examine the traversing of a given graph by using the respect to graph traversal techniques(Apply)
C21L1.5	Design a program to implement the pattern matching algorithms for the given problem.(Create)
C21L1.6	Apply data structures in the real time applications

MappingofCourseOutcomes(CO's)withPO's:

CO	PO											
	PO1	**PO2**	**PO3**	**PO4**	**PO5**	**PO6**	**PO7**	**PO8**	**PO9**	**PO10**	**PO11**	**PO12**
C21L1.1	3	2	3	3	1	1	-	-	-	-	3	1
C21L1.2	3	3	3	3	-	-	-	-	-	-	-	1
C21L1.3	3	3	3	3	-	-	-	-	1	-	1	1

C21L1.4	3	2	3	2	-	1	-	-	-	-	-	1
C21L1.5	3	3	3	3	-	-	-	-	1	-	-	2
C21L1.6	3	3	3	3	-	-	-	-	-	-	-	1
	3	**2.6**	**3**	**2.6**	**-**	**1**	**-**	**-**	**1**	**-**	**3**	**1.5**

3:High 2.Medium 1. Low

MappingofCourseOutcomes(CO's)withPSO's:

COs	PSO1	PSO2	PSO3
C21L3.1	3	3	-
C21L3.2	3	3	-
C21L3.3	3	3	2
C21L3.4	3	3	3
C21L3.5	3	3	2
C21L3.6	3	3	-
	3	3	3

SRI INDU COLLEGE OF ENGINEERING & TECHNOLOGY
(An Autonomous Institution under UGC, New Delhi)

B.Tech II year – I Sem **L T P C**
0 0 3 1.5

(R20CSE21L1) Data Structures Lab

Course Objectives:

- It covers various concepts of C programming language
- It introduces searching and sorting algorithms
- It provides an understanding of data structures such as stacks and queues.

Course Outcomes:

- Ability to develop C programs for computing and real-life applications using basic elements like control statements, arrays, functions, pointers and strings, and data structures like stacks, queues and linked lists.
- Ability to Implement searching and sorting algorithms

LIST OF EXPERIMENTS

1. Write a program that uses functions to perform the following operations on singly linked list.:
i) Creation ii) Insertion iii) Deletion iv) Traversal

2. Write a program that uses functions to perform the following operations on doubly linked list.:
i) Creation ii) Insertion iii) Deletion iv) Traversal

3. Write a program that uses functions to perform the following operations on circular linked list.:
i) Creation ii) Insertion iii) Deletion iv) Traversal

4. Write a program that implement stack (its operations) using
i) Arrays ii) Pointers

5. Write a program that implement Queue (its operations) using
i) Arrays ii) Pointers

6. Write a program that implements the following sorting methods to sort a given list of integers in ascending order
i) Bubble sort ii) Selection sort iii) Insertion sort

7. Write a program that use both recursive and non recursive functions to perform the following searching operations for a Key value in a given list of integers:
i) Linear search ii) Binary search

8. Write a program to implement the tree traversal methods.

9. Write a program to implement the graph traversal methods.

TEXTBOOKS:

1. Fundamentals of Data Structures in C, 2nd Edition, E. Horowitz, S. Sahni and Susan Anderson Freed, Universities Press.
2. Data Structures using C – A. S. Tanenbaum, Y. Langsam, and M. J. Augenstein, PHI/Pearson Education.

REFERENCE:

1. Data Structures: A Pseudocode Approach with C, 2nd Edition, R. F. Gilberg and B. A. Forouzan, Cengage Learning

	SRI INDU COLLEGE OF ENGG & TECH LAB PLAN Regulation: R20 Department of Computer Science and Engineering	Prepared on Rev1: Page: 1 of 4
	Sub. Code & Title	(R20CSE21L1) Data Structures
	Academic Year: 2022-23	Year/Sem./Section: II/I/A&B&C&D
	Faculty Name & Designation	1. Mrs. APARNA 2. Mrs. RANGAMMA 3. Mrs. KIRANMAI 4. Mrs. SAMPOORNA

Lab Plan

2022-23 II Year –I Semester CSE

S No	Topics	No. of weeks
1.	Write a program that uses functions to perform the following operations on singly linked list.: i) Creation ii) Insertion iii) Deletion iv) Traversal	1
2.	Write a program that uses functions to perform the following operations on doubly linked list.: i) Creation ii) Insertion iii) Deletion iv) Traversal	1
3.	Write a program that uses functions to perform the following operations on circular linked list.: i) Creation ii) Insertion iii) Deletion iv) Traversal	1
4.	Write a program that implement stack (its operations) using i) Arrays ii) Pointers	1
5.	Write a program that implement Queue (its operations) using i) Arrays ii) Pointers	1
6.	Write a program that implements the following sorting methods to sort a given list of integers in ascending order i) Bubble sort ii) Selection sort iii) Insertion sort.	1
7.	Write a program that use both recursive and non recursive functions to perform the following	1

	searching operations for a Key value in a given list of integers: i) Linear search ii) Binary search	
8.	Write a program to implement the tree traversal methods.	1
9.	Write a program to implement the graph traversal methods.	1

ComputerScience&Engineering

Lab Manual

Data Structures

1. Write a program that uses functions to perform the following operations on singly linked list.:
i) Creation ii) Insertion iii) Deletioniv) Traversal

2. Write a program that uses functions to perform the following operations on doubly linked list.:
i) Creation ii) Insertion iii) Deletion iv) Traversal

3. Write a program that uses functions to perform the following operations on circular linked list.:
i) Creation ii) Insertion iii) Deletion iv) Traversal

4. Write a program that implement stack (its operations) using
i) Arrays ii) Pointers

5. Write a program that implement Queue (its operations) using
i) Arrays ii) Pointers

6. Write a program that implements the following sorting methods to sort a given list of integers in ascending order
i) Bubble sort ii) Selection sort iii) Insertion sort

7. Write a program that use both recursive and non recursive functions to perform the following searching operations for a Key value in a given list of integers:
i) Linear search ii) Binary search

8. Write a program to implement the tree traversal methods.

9. Write a program to implement the graph traversal methods.

PROGRAMS

Week1.

Aim: Write a program that uses functions to perform the following operations on singly linked list.:
i) Creation ii) Insertion iii) Deletion iv) Traversal

SourceCode:

```
# include <stdio.h>
# include <conio.h>
# include <stdlib.h>
struct slinklist
{
int data;
struct slinklist *next;
};
typedef struct slinklist node;
node *start = NULL;
int menu()
{
int ch;
clrscr();
printf("\n 1.Create a list ");
printf("\n.....");
printf("\n 2.Insert a node at beginning ");
printf("\n 3.Insert a node at end");
printf("\n 4.Insert a node at middle");
printf("\n.....");
printf("\n 5.Delete a node from beginning");
printf("\n 6.Delete a node from Last");
printf("\n 7.Delete a node from Middle");
printf("\n.....");
printf("\n 8.Traverse the list (Left to Right)");
printf("\n 9.Traverse the list (Right to Left)");
printf("\n.....");
printf("\n 10. Count nodes ");
printf("\n 11. Exit ");
printf("\n\n Enter your choice: ");
scanf("%d",&ch);
return ch;
}

node* getnode()
{
node * newnode;
newnode = (node *) malloc(sizeof(node));
printf("\n Enter data: ");
scanf("%d", &newnode -> data);
newnode -> next = NULL;
return newnode;
}
```

```
int countnode(node *ptr)
{
int count=0;
while(ptr != NULL)
{
count++;
ptr = ptr -> next;
}
return (count);     }
void createlist(int n)
{
            int i;
node *newnode;
node *temp;
for(i = 0; i < n; i++)
{
newnode = getnode();
if(start == NULL)
{
start = newnode;
}
else
{
temp = start;
while(temp -> next != NULL)
temp = temp -> next;
temp -> next = newnode;
}
}
}

void traverse()
{
node *temp;
temp = start;
printf("\n The contents of List (Left to Right): \n");
if(start == NULL)
{
printf("\n Empty List");
return;
}
else
{

while(temp != NULL)
{
printf("%d-->", temp -> data);
temp = temp -> next;
}
}
printf(" X ");
}

      void rev_traverse(node *start)
```

```
{
        if(start == NULL)
        {
        return;
        }
        else
        {
        rev_traverse(start -> next);
        printf("%d -->", start -> data);
        }
}

void insert_at_beg()
{
        node *newnode;
        newnode = getnode();
        if(start == NULL)
        {
        start = newnode;
        }
        else
        {
        newnode -> next = start;
        start = newnode;
        }
}

void insert_at_end()
{
        node *newnode, *temp;
        newnode = getnode();
        if(start == NULL)
        {
        start = newnode;
        }
        else
        {
        temp = start;
        while(temp -> next != NULL)
        temp = temp -> next;
        temp -> next = newnode;
        }
}

void insert_at_mid()
{
        node *newnode, *temp, *prev;
        int pos, nodectr, ctr = 1;
        newnode = getnode();
        printf("\n Enter the position: ");
        scanf("%d", &pos);
        nodectr = countnode(start);

        if(pos > 1 && pos < nodectr)
        {
```

```
        temp = prev = start;
        while(ctr < pos)
        {
        prev = temp;
        temp = temp -> next;
        ctr++;
        }
        prev -> next = newnode;
        newnode -> next = temp;
        }
else
        printf("position %d is not a middle position", pos);
        }

        void delete_at_beg()
        {
        node *temp;
        if(start == NULL)
        {
        printf("\n No nodes are exist..");
                return ;
        }
        else
        {
        temp = start;
        start = temp -> next;
        free(temp);
        printf("\n Node deleted ");
        }
}

void delete_at_last()
{
        node *temp, *prev;
        if(start == NULL)
        {
        printf("\n Empty List..");
return ;
        }
        else
        {
        temp = start;
                prev = start;
                while(temp -> next != NULL)
                {
                prev = temp;
                temp = temp -> next;
                }
        prev -> next = NULL;
        free(temp);
        printf("\n Node deleted ");
        }
}

void delete_at_mid()
```

```
{
        int ctr = 1, pos, nodectr;
        node *temp, *prev;
        if(start == NULL)
        {
        printf("\n Empty List..");

        return ;
        }
        else
{
        printf("\n Enter position of node to delete: ");
                scanf("%d", &pos);
                nodectr = countnode(start);
                if(pos > nodectr)
                {
                printf("\nThisnode doesnot exist");
                }
                if(pos > 1 && pos < nodectr)
                {
                temp = prev = start;
                while(ctr < pos)
                {
                prev = temp;
                temp = temp -> next;
                ctr ++;
                }
                prev -> next = temp -> next;
                free(temp);
        printf("\n Node deleted..");
                }
                else
                {
                        printf("\n Invalid position..");
                        getch();
                }
}
}


void main(void)
{
int ch, n;
clrscr();
        while(1)
        {
        ch = menu();
        switch(ch)
        {
        case 1:
                if(start == NULL)
{
                        printf("\n Number of nodes you want to create: ");
                        scanf("%d", &n);
                        createlist(n);
```

```
                printf("\n List created..");
            }
            else
                printf("\n List is already created..");
            break;
    case 2:
    insert_at_beg();
            break;
    case 3:
            insert_at_end();
            break;
    case 4:
            insert_at_mid();
            break;

    case 5:
            delete_at_beg();
            break;
    case 6:
            delete_at_last();
            break;
    case 7:
            delete_at_mid();
            break;
    case 8:
            traverse();
            break;
    case 9:
            printf("\n The contents of List (Right to Left): \n");
            rev_traverse(start);
            printf(" X ");
            break;
    case 10:
            printf("\n No of nodes : %d ", countnode(start));
            break;
    case 11 :
            exit(0);
    }
    getch();
}
}
```

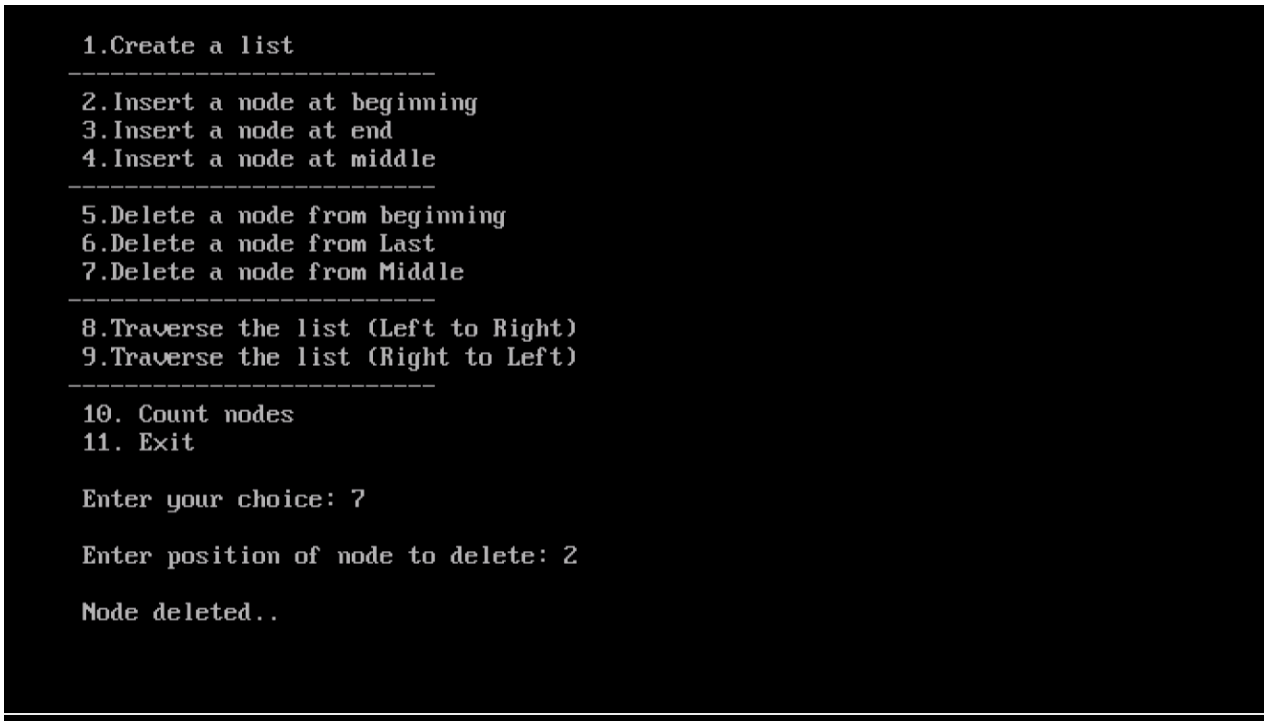
OUTPUT :

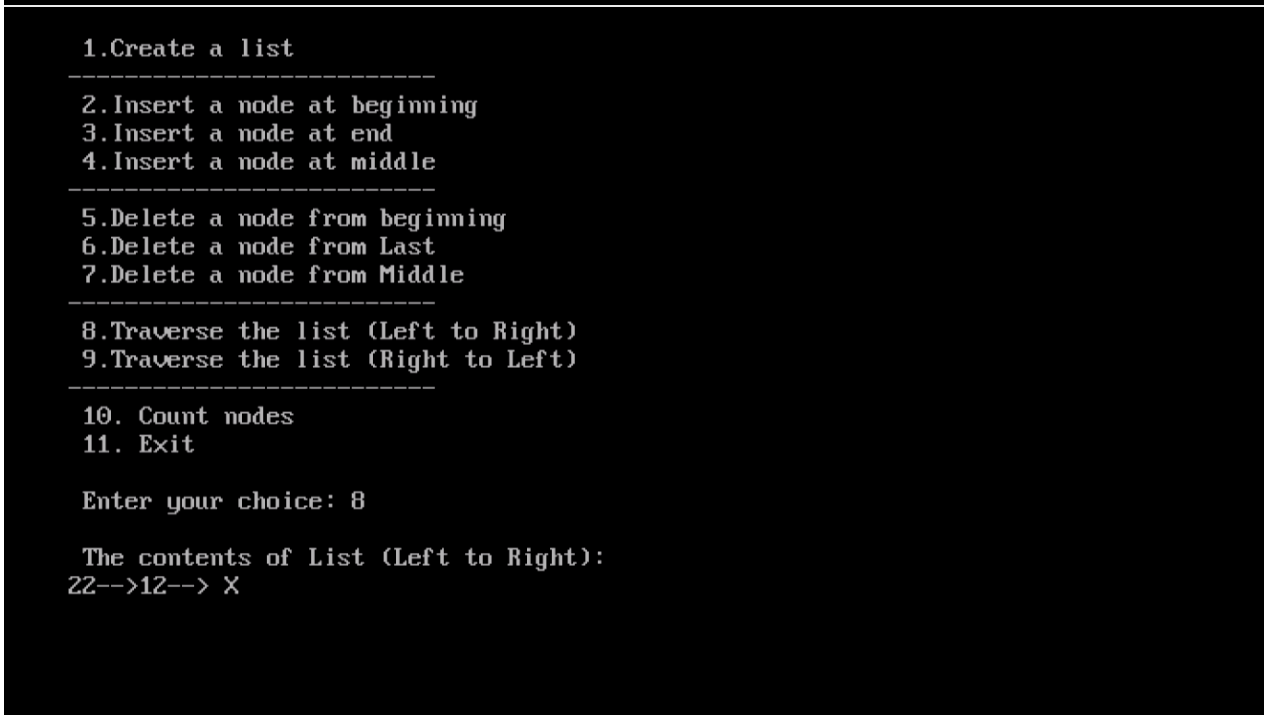
```
 1.Create a list
-------------------------
 2.Insert a node at beginning
 3.Insert a node at end
 4.Insert a node at middle
-------------------------
 5.Delete a node from beginning
 6.Delete a node from Last
 7.Delete a node from Middle
-------------------------
 8.Traverse the list (Left to Right)
 9.Traverse the list (Right to Left)
-------------------------
 10. Count nodes
 11. Exit

 Enter your choice: 1

 Number of nodes you want to create: 2

 Enter data: 11

 Enter data: 12

 List created..
```

```
 1.Create a list
-------------------------
 2.Insert a node at beginning
 3.Insert a node at end
 4.Insert a node at middle
-------------------------
 5.Delete a node from beginning
 6.Delete a node from Last
 7.Delete a node from Middle
-------------------------
 8.Traverse the list (Left to Right)
 9.Traverse the list (Right to Left)
-------------------------
 10. Count nodes
 11. Exit

 Enter your choice: 8

 The contents of List (Left to Right):
11-->12--> X
```

```
 1.Create a list
--------------------------
 2.Insert a node at beginning
 3.Insert a node at end
 4.Insert a node at middle
--------------------------
 5.Delete a node from beginning
 6.Delete a node from Last
 7.Delete a node from Middle
--------------------------
 8.Traverse the list (Left to Right)
 9.Traverse the list (Right to Left)
--------------------------
 10. Count nodes
 11. Exit

 Enter your choice: 2

 Enter data: 22
_
```

```
 1.Create a list
--------------------------
 2.Insert a node at beginning
 3.Insert a node at end
 4.Insert a node at middle
--------------------------
 5.Delete a node from beginning
 6.Delete a node from Last
 7.Delete a node from Middle
--------------------------
 8.Traverse the list (Left to Right)
 9.Traverse the list (Right to Left)
--------------------------
 10. Count nodes
 11. Exit

 Enter your choice: 8

 The contents of List (Left to Right):
22-->11-->12--> X
```





Week I Viva Questions

1. **Define self referential structure and give one example**
2. **Draw one example node of single linked list**
3. **What is NULL?**
4. **List out the operations on linked list.**
5. **Differentiate Array and linked list.**

Week2:

Aim: Write a program that uses functions to perform the following operations on doubly linked list.:
i) Creation ii) Insertion iii) Deletion iv) Traversal

SourceCode:

```
#include  <stdio.h>
#include <stdlib.h>
#include <conio.h>
struct dlinklist
{
struct dlinklist *left;
int data;
struct dlinklist *right;
};
typedef struct dlinklist node;
node *start = NULL;

node* getnode()
{
node * newnode;
newnode = (node *) malloc(sizeof(node));
printf("\n Enter data: ");
scanf("%d", &newnode -> data);
newnode -> left = NULL;
newnode -> right = NULL;
return newnode;
}
int countnode(node *start)
{
if(start == NULL)
return 0;
else
return 1 + countnode(start -> right);
}
int menu()
{
int ch;
clrscr();
printf("\n 1.Create");
printf("\n.....");
printf("\n 2. Insert a node at beginning ");
printf("\n 3. Insert a node at end");
printf("\n 4. Insert a node at middle");
printf("\n.....");
printf("\n 5. Delete a node from beginning");
printf("\n 6. Delete a node from Last");
printf("\n 7. Delete a node from Middle");
printf("\n.....");
printf("\n 8. Traverse the list from Left to Right ");
printf("\n 9. Traverse the list from Right to Left ");
printf("\n.....");
```

```
printf("\n 10.Count the Number of nodes in the list");
printf("\n 11.Exit");
printf("\n\n Enter your choice: ");
scanf("%d", &ch);
return ch;
}
void createlist(int n)
{
int i;
node *newnode;
node *temp;
for(i = 0; i < n; i++)
{
newnode = getnode();
if(start == NULL)
start = newnode;
else
{
temp = start;
while(temp -> right)
temp = temp -> right;
temp -> right = newnode;
newnode -> left = temp;
}
}
}


void traverse_left_to_right()
{
node *temp;
temp = start;
printf("\n The contents of List: ");
if(start == NULL )
printf("\n Empty List");
else
{
while(temp != NULL)
{
printf("\t %d ", temp -> data);
temp = temp -> right;
}
}
}
void traverse_right_to_left()
{
node *temp;
temp = start;
printf("\n The contents of List: ");
if(start == NULL)
printf("\n Empty List");
else
{
while(temp -> right != NULL)
```

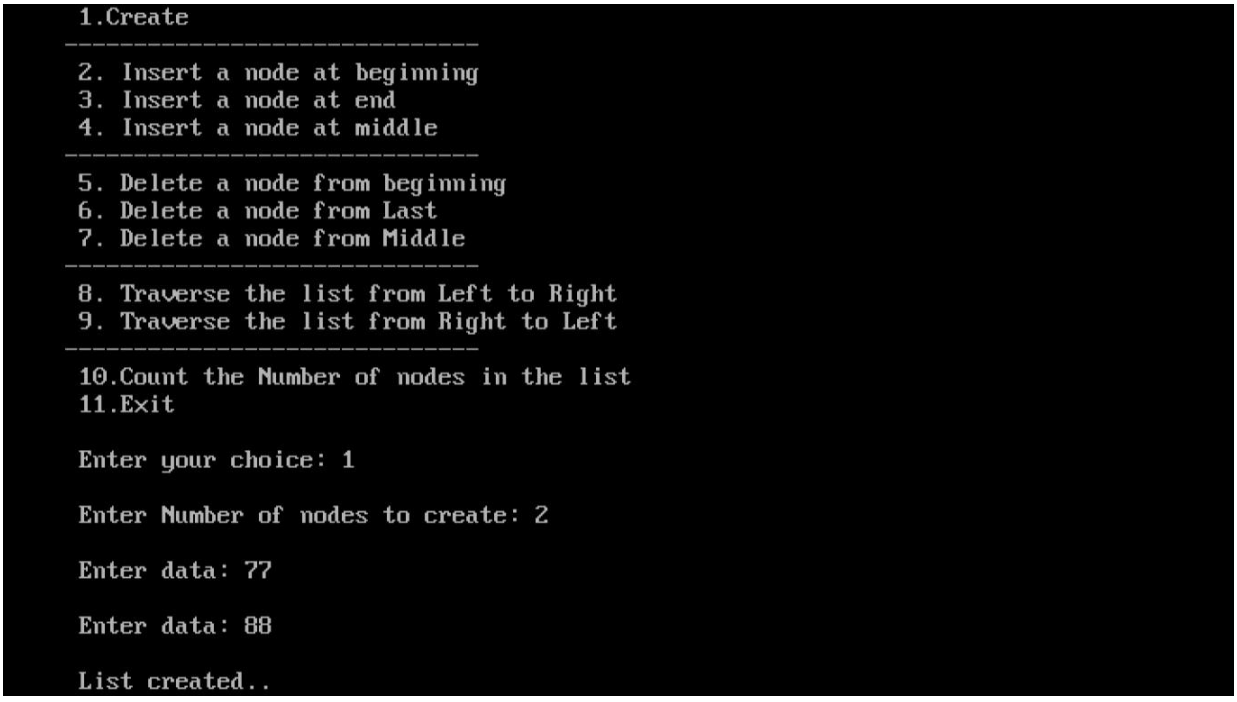
```
temp = temp -> right;
}
while(temp != NULL)
{
printf("\t%d", temp -> data);
temp = temp -> left;
}
}
void dll_insert_beg()
{
node *newnode;
newnode = getnode();
if(start == NULL)
start = newnode;
else
{
newnode -> right = start;
start -> left = newnode;
start = newnode;
}
}
void dll_insert_end()
{
node *newnode, *temp;
newnode = getnode();
if(start == NULL)
start = newnode;
else
{
temp = start;
while(temp -> right != NULL)
temp = temp -> right;
temp -> right = newnode;
newnode -> left = temp;
}
}
void dll_insert_mid()
{
node *newnode,*temp;
int pos, nodectr, ctr = 1;
newnode = getnode();
printf("\n Enter the position: ");
scanf("%d", &pos);
nodectr = countnode(start);
if(pos - nodectr >= 2)
{
printf("\n Position is out of range..");
return;
}
if(pos > 1 && pos < nodectr)
{
temp = start;
while(ctr < pos - 1)
{
```

```
temp = temp -> right;
ctr++;
}
newnode -> left  = temp;
newnode -> right = temp -> right;
temp -> right -> left = newnode;
temp -> right = newnode;
}
else
printf("position %d of list is not a middle position ", pos);
}
void dll_delete_beg()
{
node *temp;
if(start == NULL)
{
printf("\n Empty list");
getch();
return ;
}
else
{
temp = start;
start = start -> right;
start -> left = NULL;
free(temp);
}
}
void dll_delete_last()
{
node *temp;
if(start == NULL)
{
printf("\n Empty list");
getch();
return ;
}
else
{
temp = start;
while(temp -> right != NULL)
temp = temp -> right;
temp -> left -> right = NULL;
free(temp);
temp = NULL;
}
}
void dll_delete_mid()
{
int i = 0, pos, nodectr;
node *temp;
if(start == NULL)
{
printf("\n Empty List");
```

```
getch();
return;
}
else
{
printf("\n Enter the position of the node to delete: ");
scanf("%d", &pos);
nodectr = countnode(start);
if(pos > nodectr)
{
printf("\nthis node does not exist");
getch();
return;
}
if(pos > 1 && pos < nodectr)
{
temp = start;
i= 1;
while(i < pos)
{
temp = temp -> right;
i++;
}
temp -> right -> left = temp -> left;
temp -> left -> right = temp -> right;
free(temp);
printf("\n node deleted..");
}
else
{
printf("\n It is not a middle position..");
getch();
}
}
}
void main(void)
{
int ch, n;
clrscr();
while(1)
{
ch = menu();
switch( ch)
{
case 1 :
printf("\n Enter Number of nodes to create: ");
scanf("%d", &n);
createlist(n);

printf("\n List created..");
break;
case 2 :
dll_insert_beg();
break;
```

```
case 3 :
dll_insert_end();
break;
case 4 :
dll_insert_mid();
break;
case 5 :
dll_delete_beg();
break;
case 6 :
dll_delete_last();
break;
case 7 :
dll_delete_mid();
break;
case 8 :
traverse_left_to_right();
break;
case 9 :
traverse_right_to_left();
break;
case 10 :
printf("\n Number of nodes: %d", countnode(start));
break;
case 11:
exit(0);
}
getch();
}
}
```

OUTPUT :



```
 1.Create
-----------------------------
 2. Insert a node at beginning
 3. Insert a node at end
 4. Insert a node at middle
-----------------------------
 5. Delete a node from beginning
 6. Delete a node from Last
 7. Delete a node from Middle
-----------------------------
 8. Traverse the list from Left to Right
 9. Traverse the list from Right to Left
-----------------------------
 10.Count the Number of nodes in the list
 11.Exit

 Enter your choice: 3

 Enter data: 44
_
```

```
 1.Create
-----------------------------
 2. Insert a node at beginning
 3. Insert a node at end
 4. Insert a node at middle
-----------------------------
 5. Delete a node from beginning
 6. Delete a node from Last
 7. Delete a node from Middle
-----------------------------
 8. Traverse the list from Left to Right
 9. Traverse the list from Right to Left
-----------------------------
 10.Count the Number of nodes in the list
 11.Exit

 Enter your choice: 8

 The contents of List:   77      88      44 _
```

Week II Viva Questions

1. ***Advantage of linked lists.***
2. ***Difference between single and double linked list.***
3. ***Define node structure of double linked list.***
4. ***What is traversal?***
5. ***List types of Linked lists.***

Week3:

Aim: Write a program that uses functions to perform the following operations on circular linked list.:
i) Creation ii) Insertion iii) Deletion iv) Traversal

Sourcecode:

```
# include <stdio.h>
# include <conio.h>
# include <stdlib.h>
struct cslinklist
{
int data;
struct cslinklist *next;
};
typedef struct cslinklist node;
node *start = NULL;
int nodectr;
node* getnode()
{
node * newnode;
newnode = (node *) malloc(sizeof(node));
printf("\n Enter data: ");
scanf("%d", &newnode -> data);
newnode -> next = NULL;
return newnode;
}

int menu()
{
int ch;
clrscr();
printf("\n 1. Create a list ");
printf("\n\n.....");
printf("\n 2. Insert a node at beginning ");
printf("\n 3. Insert a node at end");
printf("\n 4. Insert a node at middle");
printf("\n\n.....");
printf("\n 5. Delete a node from beginning");
printf("\n 6. Delete a node from Last");
printf("\n 7. Delete a node from Middle");
printf("\n\n.....");
printf("\n 8. Display the list");
printf("\n 9. Exit");
printf("\n\n.....");
printf("\n Enter your choice: ");
scanf("%d", &ch);
return ch;
}
void createlist(int n)
{
int i;
node *newnode;
node *temp;
```

```
nodectr = n;
for(i = 0; i < n ; i++)
{
newnode = getnode();
if(start == NULL)
{
start = newnode;
}
else
{
temp = start;
while(temp -> next != NULL)
temp = temp -> next;
temp -> next = newnode;
}
}
newnode ->next = start; /* last node is pointing to starting node */
}
void display()
{
node *temp;
temp = start;
printf("\n The contents of List (Left to Right): ");
if(start == NULL )
printf("\n Empty List");
else
{
do
{
printf("\t %d ", temp -> data);
temp = temp -> next;
} while(temp != start);
printf(" X ");
}
}

void cll_insert_beg()
{
node *newnode, *last;
newnode = getnode();
if(start == NULL)
{
start = newnode;
newnode -> next = start;
}
else
{
last = start;
while(last -> next != start)
last= last -> next;
newnode -> next = start;
start = newnode;
last -> next = start;
}
printf("\n Node inserted at beginning..");
```

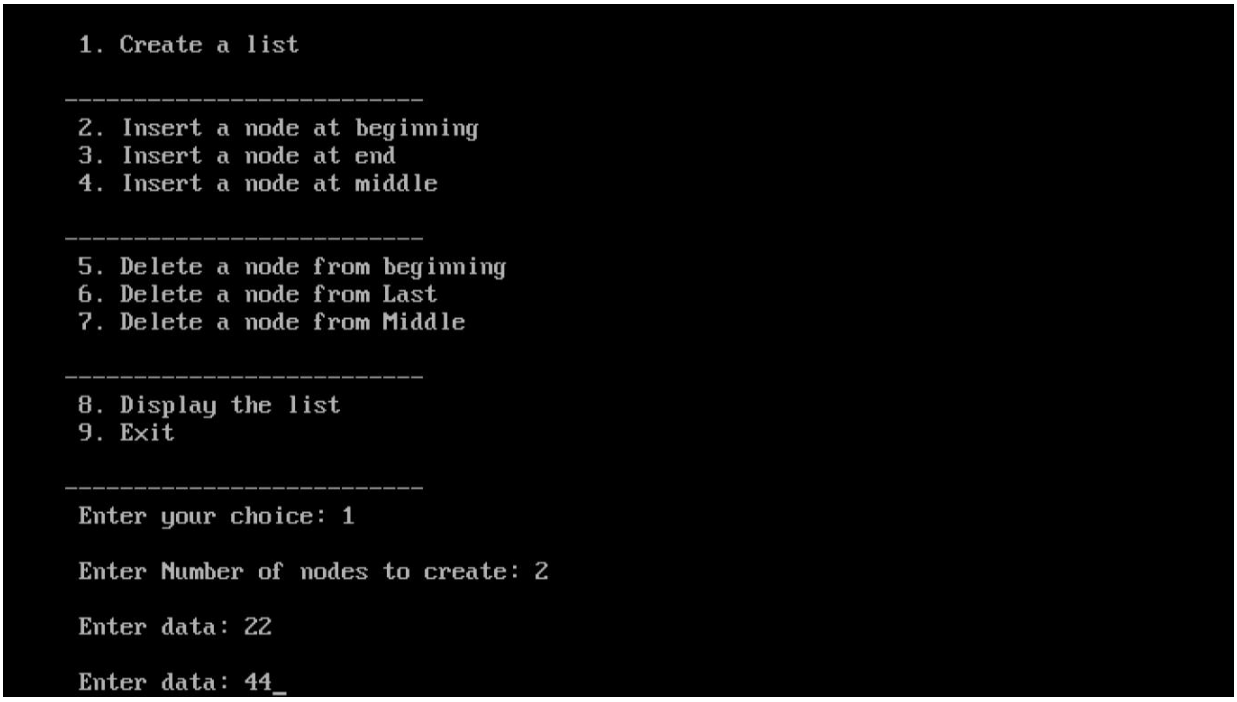
```
nodectr++;
}
void cll_insert_end()
{
node *newnode, *temp;
newnode = getnode();
if(start == NULL )
{
start = newnode;
newnode -> next = start;
}
else
{
temp = start;
while(temp -> next != start)
temp = temp -> next;
temp -> next = newnode;
newnode -> next = start;
}
printf("\n Node inserted at end..");
nodectr++;
}
void cll_insert_mid()
{
node *newnode, *temp, *prev;
int i, pos ;
newnode = getnode();
printf("\n Enter the position: ");
scanf("%d", &pos);
if(pos > 1 && pos < nodectr)
{
temp = start;
prev = temp;
i= 1;
while(i < pos)
{
prev = temp;
temp = temp -> next;
i++;
}
prev -> next = newnode;
newnode -> next = temp;

nodectr++;
printf("\n Node inserted at middle..");
}
else
{
printf("position %d of list is not a middle position ", pos);
}
}
void cll_delete_beg()
{
node *temp, *last;
if(start == NULL)
```

```
{
printf("\n No nodes exist..");
getch();
return ;
}
else
{
last = temp = start;
while(last -> next != start)
last= last -> next;
start = start -> next;
last -> next = start;
free(temp);
nodectr--;
printf("\n Node deleted..");
if(nodectr == 0)
start = NULL;
}
}
void cll_delete_last()
{
node *temp,*prev;
if(start == NULL)
{
printf("\n No nodes exist..");
getch();
return ;
}
else
{
temp = start;
prev = start;
while(temp -> next != start)
{
prev = temp;
temp = temp -> next;
}
prev -> next = start;
free(temp);
nodectr--;
if(nodectr == 0)
start = NULL;
printf("\n Node deleted..");
}
}

void cll_delete_mid()
{
int i = 0, pos;
node *temp, *prev;
if(start == NULL)
{
printf("\n No nodes exist..");
getch();
return ;
```

```
}
else
{
printf("\n Which node to delete: ");
scanf("%d", &pos);
if(pos > nodectr)
{
printf("\nThis node does not exist");
getch();
return;
}
if(pos > 1 && pos < nodectr)
{
temp=start;
prev = start;
i= 0;
while(i < pos - 1)
{
prev = temp;
temp = temp -> next ;
i++;
}
prev -> next = temp -> next;
free(temp);
nodectr--;
printf("\n Node Deleted..");
}
else
{
printf("\n It is not a middle position..");
getch();
}
}
}
void main(void)
{
int result;
int ch, n;
clrscr();
while(1)
{
ch = menu();
switch(ch)
{
case 1 :
if(start == NULL)
{
printf("\n Enter Number of nodes to create: ");
scanf("%d", &n);
createlist(n);
printf("\nList created..");
}

else
printf("\n List is already Exist..");
```

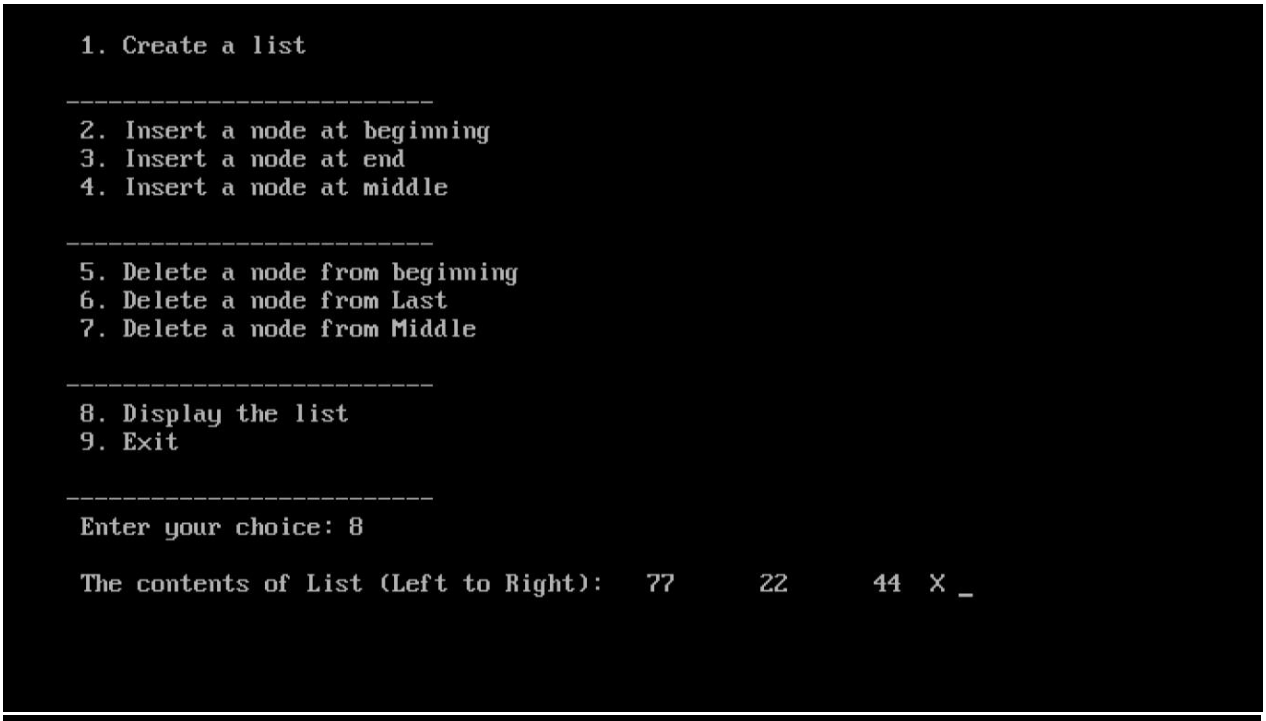
```
break;
case 2 :
cll_insert_beg();
break;
case 3 :
cll_insert_end();
break;
case 4 :
cll_insert_mid();
break;
case 5 :
cll_delete_beg();
break;
case 6 :
cll_delete_last();
break;
case 7 :
cll_delete_mid();
break;
case 8 :
display();
break;
case 9 :
exit(0);
}
getch();
}
 }
```

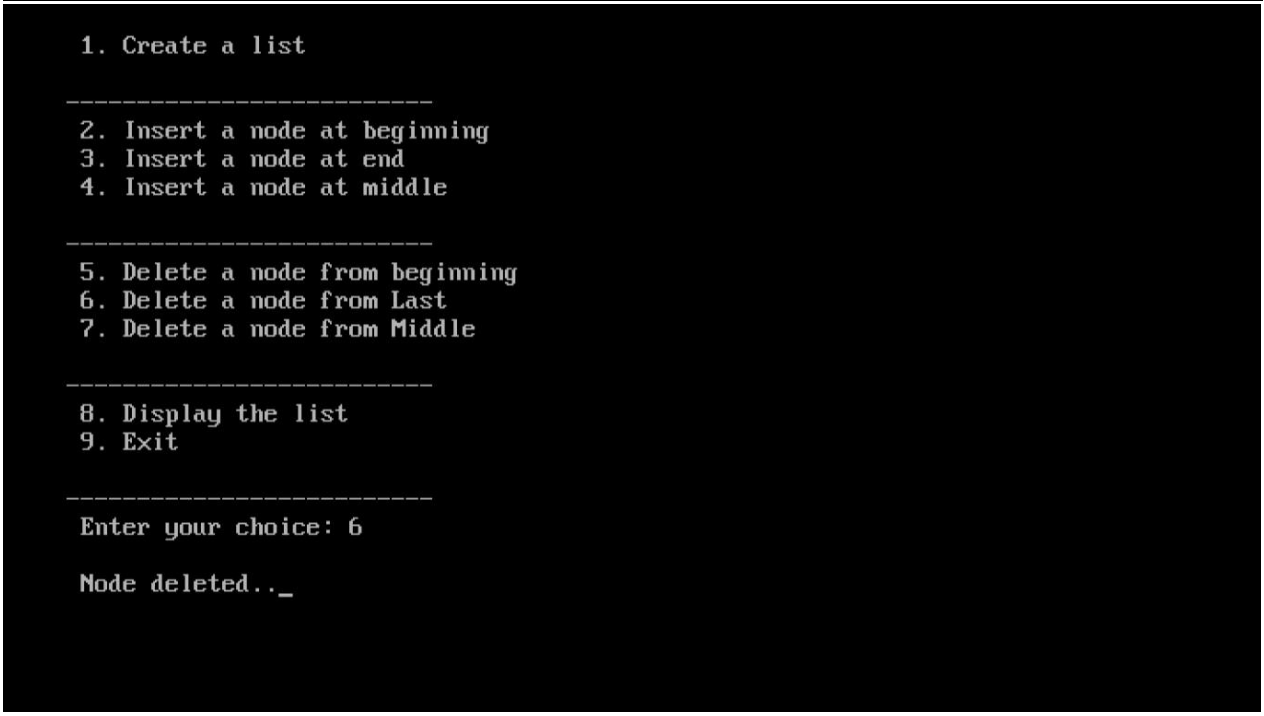
Output :



```
 1. Create a list

--------------------------
 2. Insert a node at beginning
 3. Insert a node at end
 4. Insert a node at middle

--------------------------
 5. Delete a node from beginning
 6. Delete a node from Last
 7. Delete a node from Middle

--------------------------
 8. Display the list
 9. Exit

--------------------------
 Enter your choice: 2

 Enter data: 77

 Node inserted at beginning..
```

```
 1. Create a list

--------------------------
 2. Insert a node at beginning
 3. Insert a node at end
 4. Insert a node at middle

--------------------------
 5. Delete a node from beginning
 6. Delete a node from Last
 7. Delete a node from Middle

--------------------------
 8. Display the list
 9. Exit

--------------------------
 Enter your choice: 8

 The contents of List (Left to Right):   77      22      44  X
```





Viva Questions

1. Draw an example to insert node in Circular linked list.
2. Write the Advantage of Circular linked list.
3. Differentiate single , double ,circular lists.
4. List applications of linked list.
5. What is while(1)?

Week4:

Aim: Write a program that implement stack (its operations) using
i)Arrays ii) Pointers

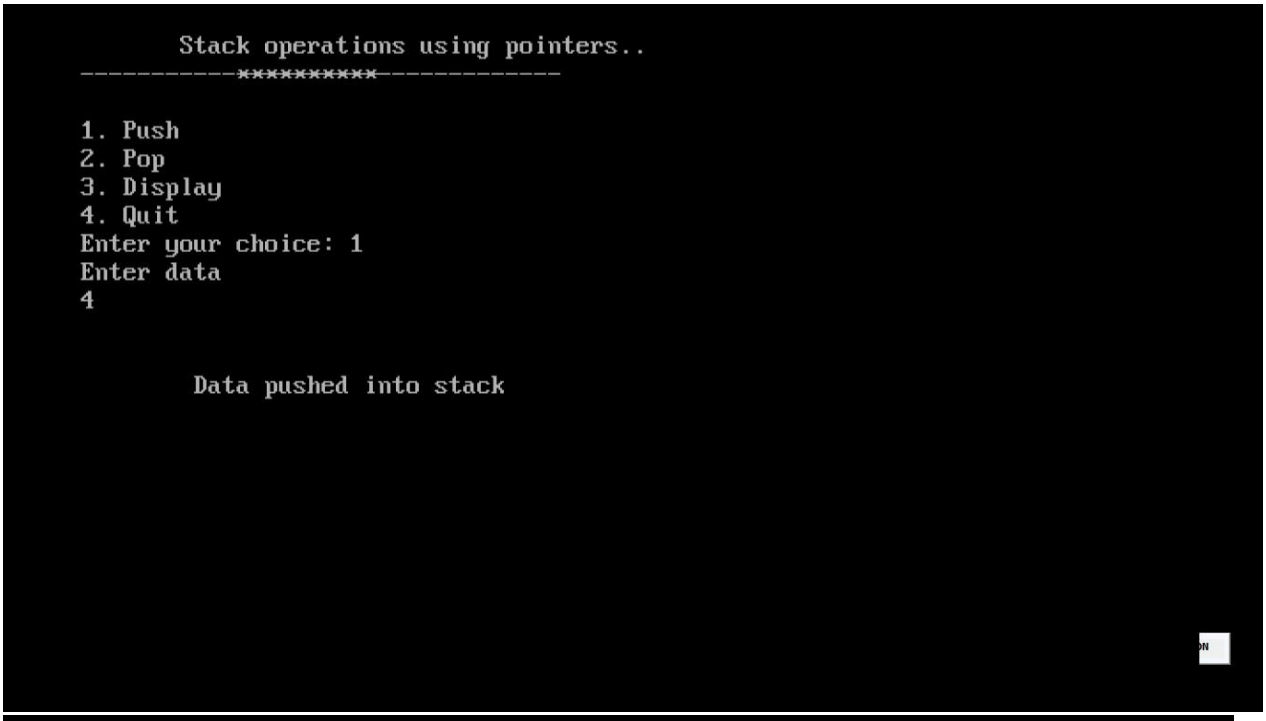
Source code to implement Stack using linked list :

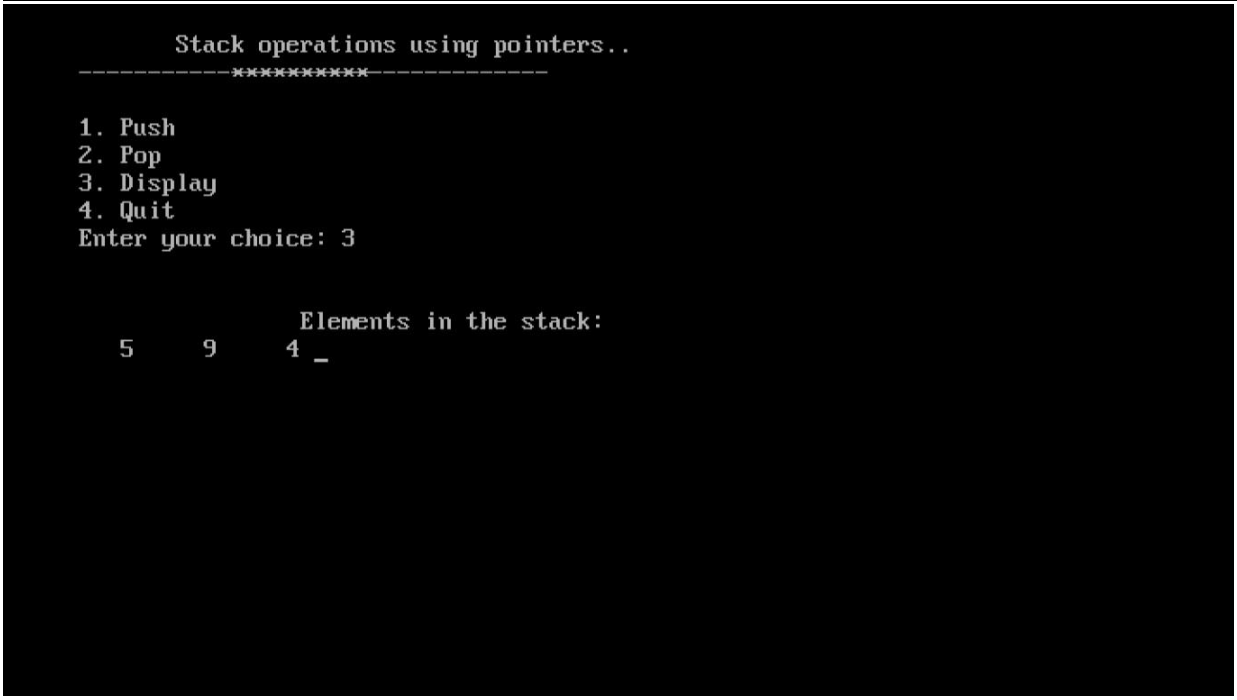
```
# include <stdio.h>
# include <conio.h>
# include <stdlib.h>
struct stack
{
int data;
struct stack *next;
};
void push();
void pop();
void display();
typedef struct stack node;
node *start=NULL;
node *top = NULL;
node* getnode()
{
node *temp;
temp=(node *) malloc( sizeof(node)) ;
printf("\n Enter data ");
scanf("%d", &temp -> data);
temp -> next = NULL;
return temp;
}
void push(node *newnode)
{
node *temp;
if( newnode == NULL )
{
printf("\n Stack Overflow..");
return;
if(start == NULL)
{
start = newnode;
top = newnode;
}
else
{
temp = start;
while( temp -> next != NULL)
temp = temp -> next;
temp -> next = newnode;
top = newnode;
}
printf("\n\n\t Data pushed into stack");
```

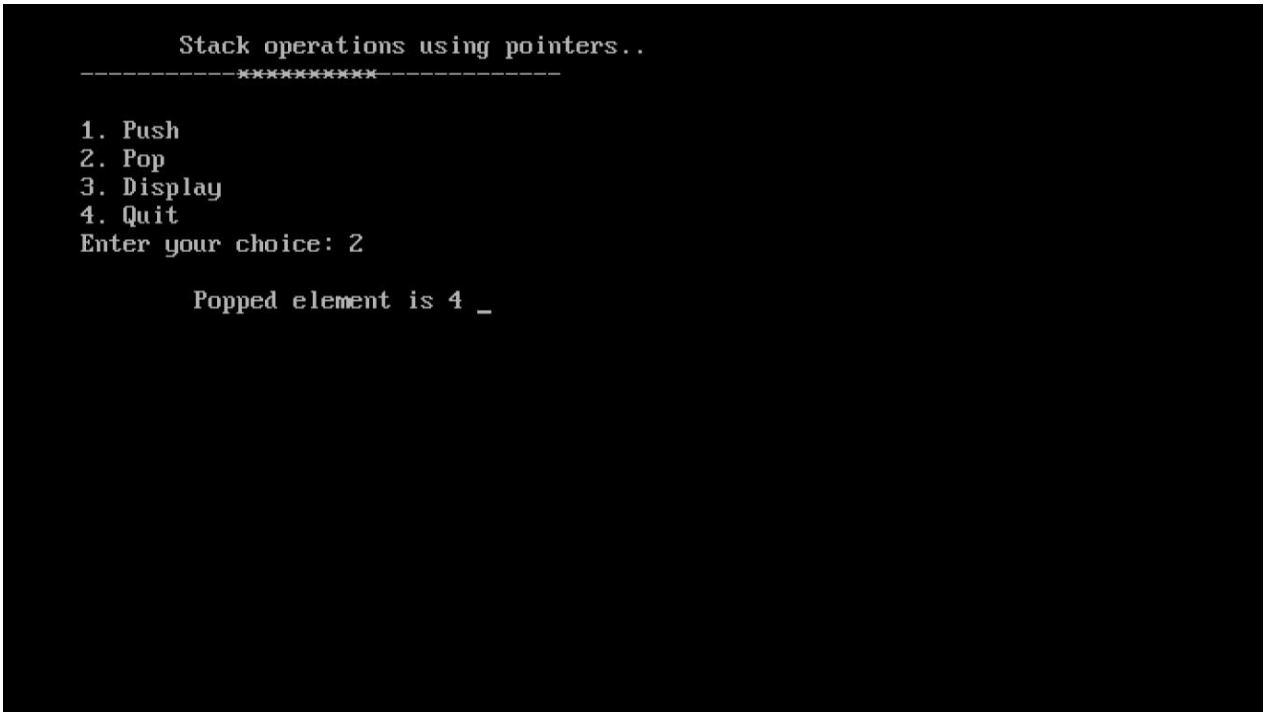
```
}
void pop()
{
node *temp;
if(top == NULL)
{
printf("\n\n\t Stack underflow");
return;
}
temp = start;
if( start -> next == NULL)
{
printf("\n\n\t Popped element is %d ", top -> data);
start = NULL;
free(top);
top = NULL;
}
else
{
while(temp -> next != top)
{
temp = temp -> next;
}
temp -> next = NULL;
printf("\n\n\t Popped element is %d ", top -> data);
free(top);
top = temp;
}
}
void display()
{
node *temp;
if(top == NULL)
{
printf("\n\n\t\t Stack is empty ");
}
else
{
temp = start;
printf("\n\n\n\t\t Elements in the stack: \n");
printf("%5d ", temp -> data);
while(temp != top)
{
temp = temp -> next;
printf("%5d ", temp -> data);
}
}
}

char menu()
{
char ch;
clrscr();
printf("\n \tStack operations using pointers.. ");
printf("\n ----------********** ----------\n");
```

```
printf("\n 1. Push ");
printf("\n 2. Pop ");
printf("\n 3. Display");
printf("\n 4. Quit ");
printf("\n Enter your choice: ");
ch = getche();
return ch;
}
void main()
{
char ch;
node *newnode;
do
{
ch = menu();
switch(ch)
{
case '1' :
        newnode = getnode();
        push(newnode);
        break;
case '2' :
        pop();
        break;
case '3' :
        display();
        break;
case '4':
        return;
}
getch();
} while( ch != '4' );
}
```

OUTPUT :


Stack operations using pointers..
1. Push
2. Pop
3. Display
4. Quit
Enter your choice: 1
Enter data
4
Data pushed into stack


Stack operations using pointers..
1. Push
2. Pop
3. Display
4. Quit
Enter your choice: 3
Elements in the stack:
5 9 4



Viva Questions

1. ***Define Stack and list operations on stack.***
2. ***What is stack overflow?***
3. ***What is stack underflow?***
4. ***List any two stack applications.***
5. ***List types of expressions.***

Week5:

Aim: Write a program that implement Queue (its operations) using

i) Arrays ii) Pointers

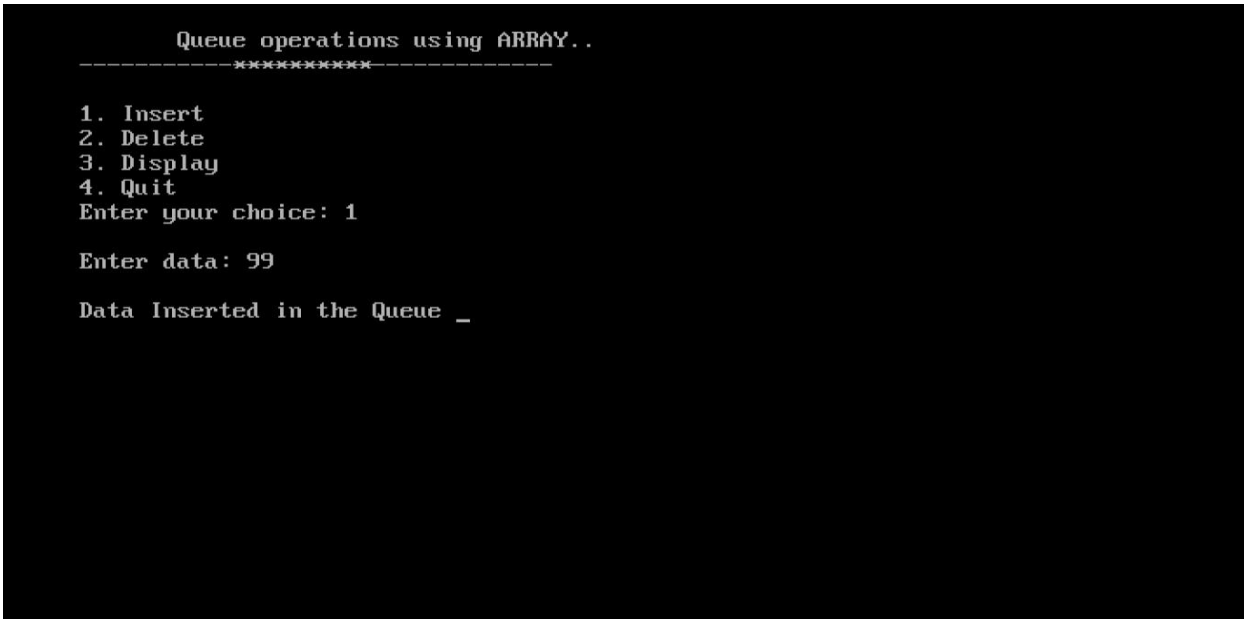
Sourcecode (Using array) :

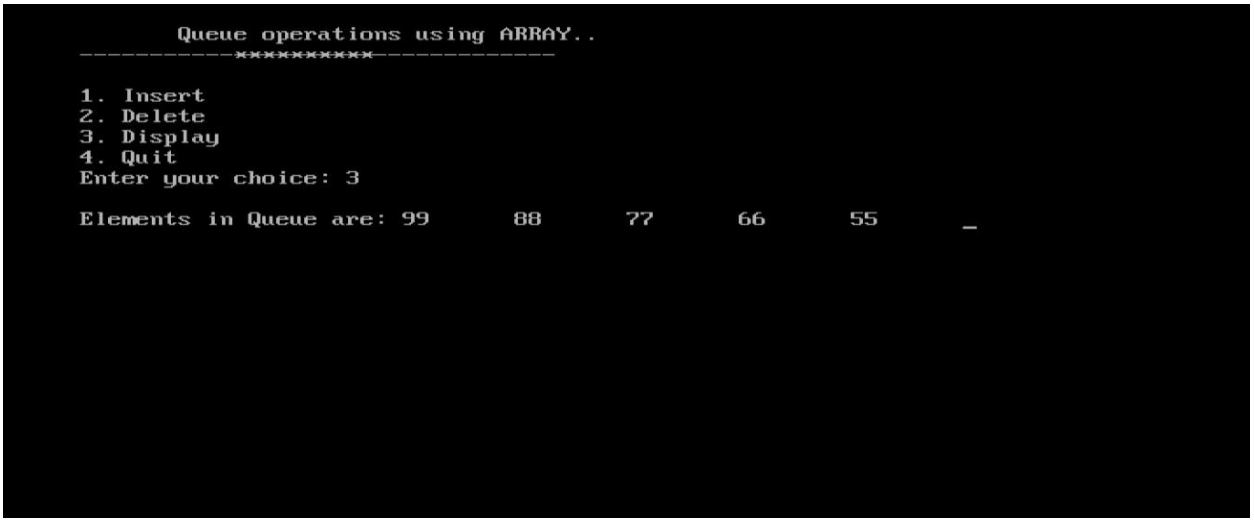
```
# include <conio.h>
# define MAX 6
int Q[MAX];
int front, rear;
void insertQ()
{
int data;
if(rear == MAX)
{
printf("\n Linear Queue is full");
return;
}
else
{
printf("\n Enter data: ");
scanf("%d", &data);
Q[rear] = data;
rear++;
printf("\n Data Inserted in the Queue ");
}
}
void deleteQ()
{
if(rear == front)
{
printf("\n\n Queue is Empty..");
return;
}
else
{
printf("\n Deleted element from Queue is %d", Q[front]);
front++;
}
}
void displayQ()
{
int i;
if(front == rear)
{
printf("\n\n\t Queue is Empty");
return;
}
else
{
printf("\n Elements in Queue are: ");
for(i = front; i < rear; i++)
```

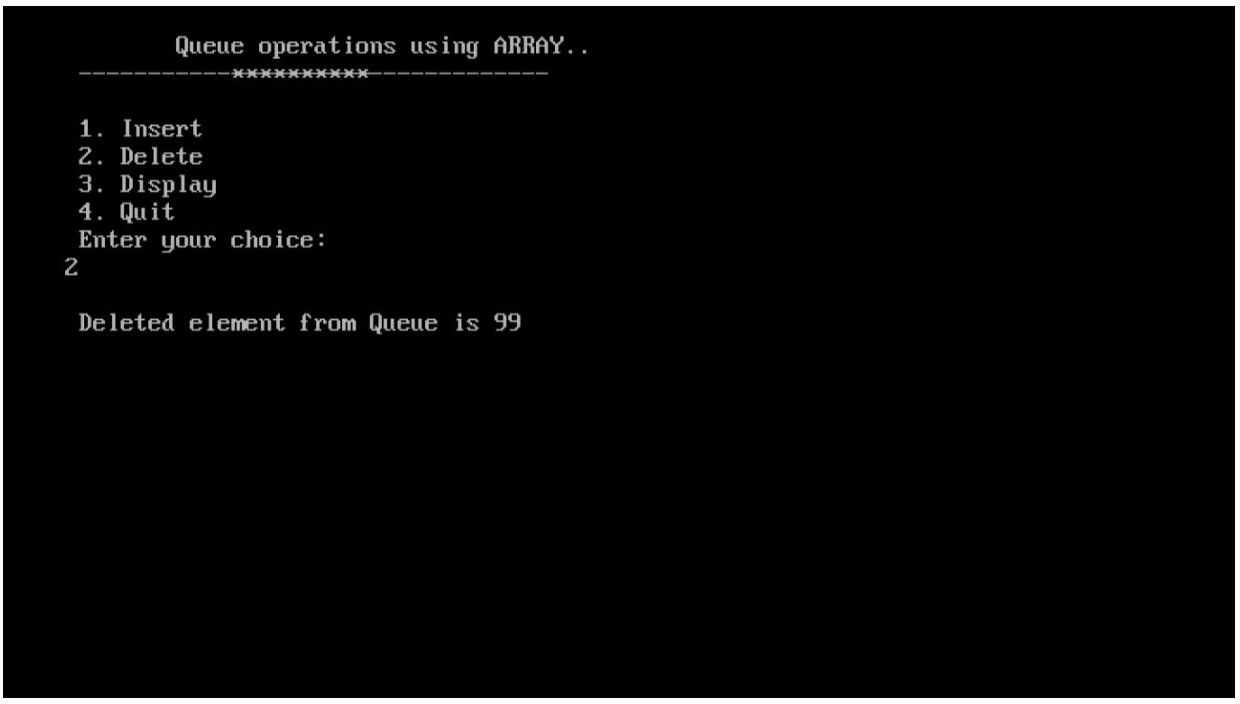
```
{
printf("%d\t", Q[i]);
}
}
}
int menu()
{
int ch;
clrscr();
printf("\n \tQueue operations using ARRAY..");
printf("\n -----********** -----\n");
printf("\n 1. Insert ");
printf("\n 2. Delete ");
printf("\n 3. Display");
printf("\n 4. Quit ");
printf("\n Enter your choice: ");
scanf("%d", &ch);
return ch;
}
void main()
{
int ch;
do
{
ch = menu();
switch(ch)
{
case 1:
insertQ();
break;
case 2:
deleteQ();
break;
case 3:
displayQ();
break;
case 4:
return;
}
getch();
} while(1);
```

}

Output :


Queue operations using ARRAY..
1. Insert
2. Delete
3. Display
4. Quit
Enter your choice: 1
Enter data: 99
Data Inserted in the Queue


Queue operations using ARRAY..
1. Insert
2. Delete
3. Display
4. Quit
Enter your choice: 3
Elements in Queue are: 99 88 77 66 55


Queue operations using ARRAY..
1. Insert
2. Delete
3. Display
4. Quit
Enter your choice:
2
Deleted element from Queue is 99

ii) Source code to implement Queue operations using pointer

```
# include <stdlib.h>
# include <conio.h>
struct queue
{
int data;
struct queue *next;
};
typedef struct queue node;
node *front = NULL;
node *rear = NULL;
node* getnode()
{
node *temp;
temp = (node *) malloc(sizeof(node)) ;
printf("\n Enter data ");
scanf("%d", &temp -> data);
temp -> next = NULL;
return temp;
}
void insertQ()
{
node *newnode;
newnode= getnode();
if(newnode== NULL)
{
printf("\n Queue Full");
return;
}
if(front == NULL)
{
front = newnode;
rear=newnode;
}
else
{
rear -> next = newnode;
rear=newnode;
}
printf("\n\n\t Data Inserted into the Queue..");
}
void deleteQ()
{
node *temp;
if(front == NULL)
{
printf("\n\n\t Empty Queue..");
return;
}
temp = front;
front = front -> next;
printf("\n\n\t Deleted element from queue is %d ", temp -> data);
free(temp);
}
```

```
void displayQ()
{
node *temp;
if(front == NULL)
{
printf("\n\n\t\t Empty Queue ");
}
else
{
temp = front;
printf("\n\n\n\t\t Elements in the Queue are: ");
while(temp != NULL)
{
printf("%5d ", temp -> data);
temp = temp -> next;
}
}
}
char menu()
{
char ch;
clrscr();
printf("\n \t..Queue operations using pointers.. ");
printf("\n\t ----------********** ---------- \n");
printf("\n 1. Insert ");
printf("\n 2. Delete ");
printf("\n 3. Display");
printf("\n 4. Quit ");
printf("\n Enter your choice: ");
ch = getche();
return ch;
}
void main()
{
char ch;
do
{
ch = menu();
switch(ch)
{
case '1' :
insertQ();
break;
case '2' :
deleteQ();
break;
case '3' :
displayQ();
break;
case '4':
return;
}
getch();
} while(ch != '4'        }
```

Viva Questions

1. ***Define Queue.***
2. ***List the condition for queue full and queue empty.***
3. ***List types of queues.***
4. ***Write the applications of queue.***
5. ***Limitation of linear queue.***

Week6:

Aim: Write a program that implements the following sorting methods to sort a given list of integers in ascending order

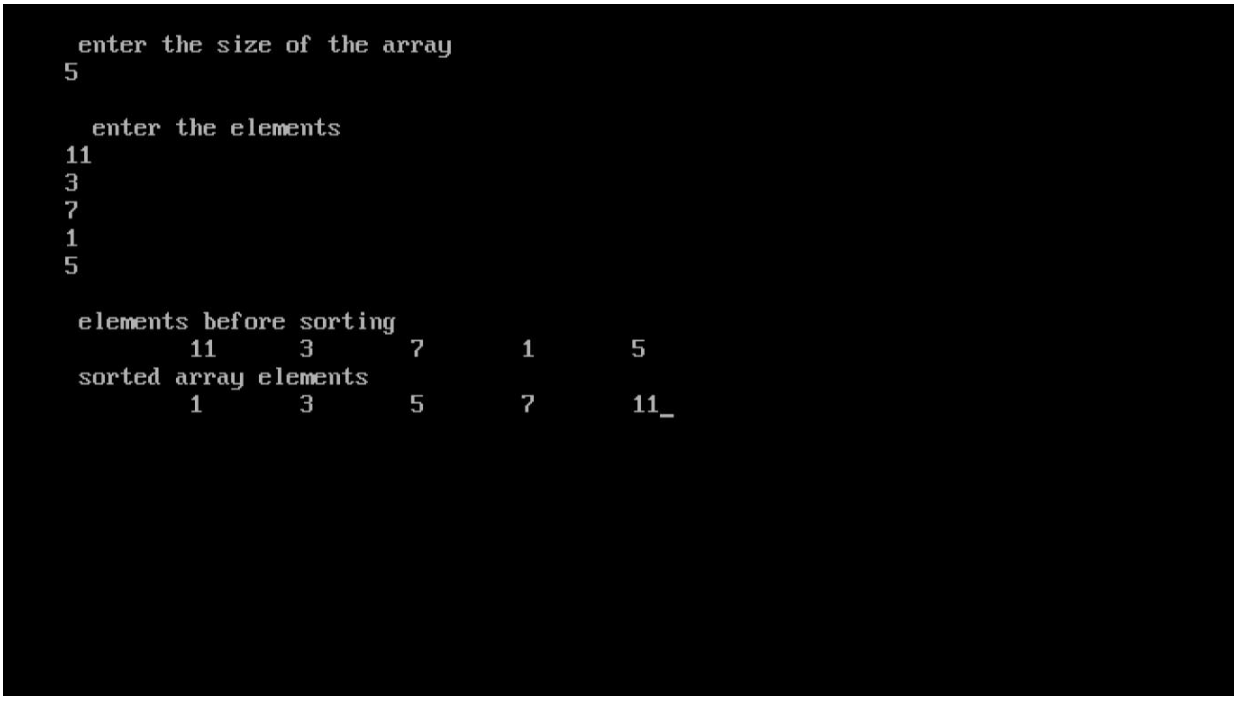
i) Bubble sort ii) Selection sort iii) Insertion sort

Source code:

i) Bubble sort

```
#include<stdio.h>
#include<conio.h>
#include<alloc.h>

 void bubblesort(int *,int);
  void main()
 {
    int *a,n,i;
     clrscr();
      printf("\n enter the size of the array \n");
       scanf("%d",&n);
           a=(int *)calloc(n,sizeof(int));
             printf("\n  enter the elements \n");
             for(i=0;i<n;i++)
             scanf("%d",&a[i]);
            printf("\n elements before sorting\n");       a
              for(i=0;i<n;i++)
                printf("\t %d",a[i]);
                      bubblesort(a,n);
                printf("\n sorted array elements \n");
                 for(i=0;i<n;i++)
                        printf("\t %d",a[i]);
             getch();
 }
   void bubblesort(int *a,int n)
    {
           int i,j,t;
            for(i=0;i<n;i++)
              {
                 for(j=0;j<n-i-1;j++)
                      {
                        if(a[j]>a[j+1])
                         {
                           t=a[j];
                           a[j]=a[j+1];
                           a[j+1]=t;
                          }
                 }
             }
    }
```

Bubble sort Output :



ii) Selection sort

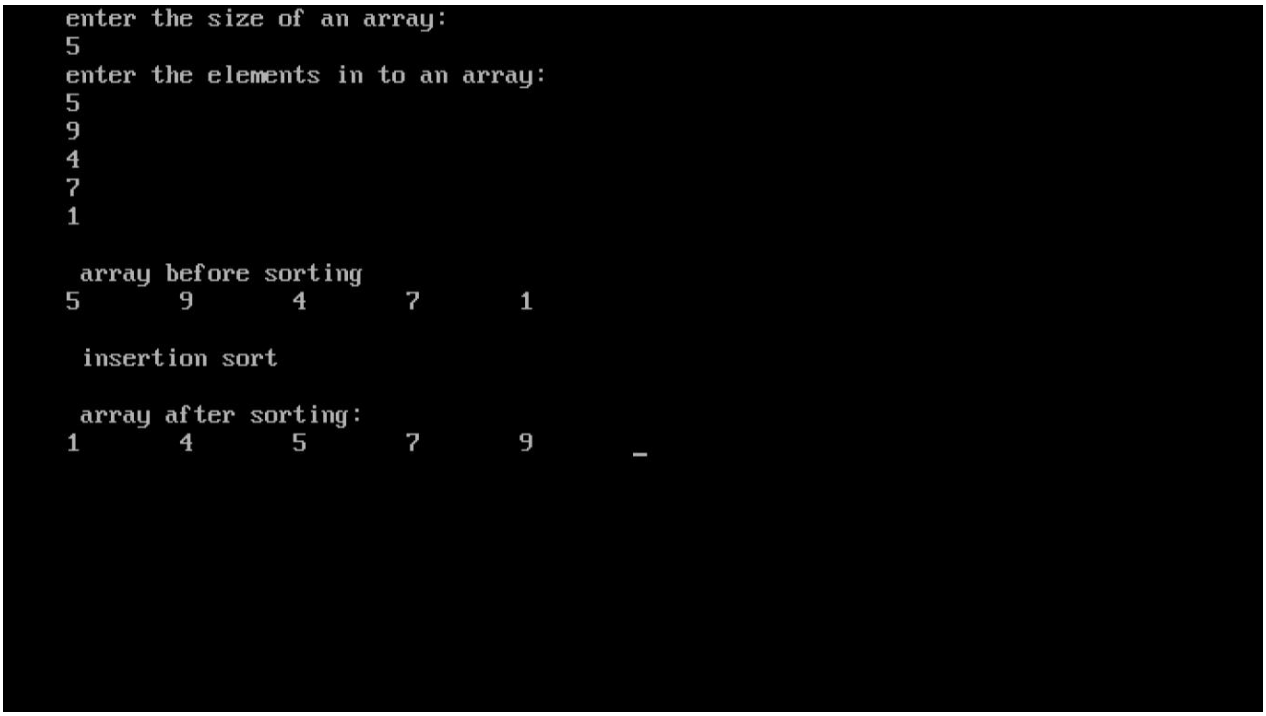
```
#include<stdio.h>
#include<conio.h>
void main()
 {
   int a[20],i,j,temp,n;
    clrscr();
     printf("\n enter the size of an array: \n");
      scanf("%d",&n);
      printf("enter array elementas \n");
       for(i=0;i<n;i++)
           scanf("%d",&a[i]);
           printf("\n selection sort: \n\n");
           printf("array elements before sorting: \n");
              for(i=0;i<n;i++)
                    printf("%d \t",a[i]);
                     for(i=0;i<=n-2;i++)
                      {
                        for(j=i+1;j<n;j++)
                         {
                              if(a[i]>a[j])
                                {
                                  temp=a[i];
                                  a[i]=a[j];
                                  a[j]=temp;
                                }
                         }
                      }
           printf("\n array after sorting: \n");
             for(i=0;i<n;i++)
              printf("%d \t",a[i]);
    getch();
 }
```

Output

```
 enter the size of an array:
6
enter array elementas
8
4
9
3
6
22

 selection sort:

array elements before sorting:
8       4       9       3       6       22
 array after sorting:
3       4       6       8       9       22
```

Insertion sort :

```
#include<stdio.h>
#include<conio.h>
void main()
  {
    int a[20],i,j,k,temp,n;
     clrscr();
      printf("enter the size of an array: \n");
       scanf("%d",&n);
           printf("enter the elements in to an array: \n");
               for(i=0;i<n;i++)
                 scanf("%d",&a[i]);
              printf("\n array before sorting \n");
                 for(i=0;i<n;i++)
                      printf("%d \t",a[i]);
                        printf("\n\n insertion sort \n");
                          for(i=1;i<n;i++)
                           {
                                for(j=0;j<i;j++)
                                  {
                                    if(a[j]>a[i])
                                     {
                                           temp=a[j];
                                           a[j]=a[i];
                                           for(k=i;k>j;k--)
                                           a[k]=a[k-1];
                                           a[k+1]=temp;
                                     }
                                  }
                           }
      printf("\n array after sorting: \n");
           for(i=0;i<n;i++)
             printf("%d \t",a[i]);
      getch();
 }
```



Viva Questions

1. Define sorting.
2. What is external sorting?
3. What is internal sorting ?
4. Compare bubble and selection sorts.
5. Give one numerical example for each sorting .

WEEK 7:

AIM : Write a program that use both recursive and non recursive functions to perform the following searching operations for a Key value in a given list of integers:

i) Linear search ii) Binary search

i) Linear Search

```
#include<stdio.h>
#include<conio.h>
void linear(int[],int,int);

 void  main()
 {
  int i,n,key;
  int a[50];
  clrscr();
   printf("\n how many elements you want to insert into an array= \n");
       scanf("%d",&n);
   printf("\n\n enter the array elements= \n");
        for(i=0;i<n;i++)
          {
           scanf("%d",&a[i]);
          }
   printf("the elements in the array=\n");
        for(i=0;i<n;i++)
          {
           printf("%5d",a[i]);
          }
    printf("\n which elements you want to search=");
        scanf("%d",&key);
          linear(a,n,key);
       getch();
}

 void linear(int a[],int n,int key)
   {
        int flag=1,i;
        for(i=0;i<n;i++)
          {
           if(a[i]==key)
            {
                printf("\n search is successfull \n");
                printf("elements %d found at location %d \n",key,i+1);
                flag=0;
                break;
            }
          }
        if(flag)
         {
          printf("\n unsuccessful search %d not found ",key);
         }
          getch();
   }
```

Output :

```
 how many elements you want to insert into an array=
6


 enter the array elements=
7
3
9
5
1
8
the elements in the array=
    7    3    9    5    1    8
 which elements you want to search=5

 search is successfull
elements 5 found at location 4
```

```
 how many elements you want to insert into an array=
6


 enter the array elements=
       7
3
9
5
1
8
the elements in the array=
    7    3    9    5    1    8
 which elements you want to search=2

 unsuccessful search 2 not found
```

ii) Binary Search

```
#include<stdio.h>
#include<conio.h>
void binary(int [],int,int);
 void main()
  {
    int a[50];
    int i,n,key;
    clrscr();
    printf("\n enter how many elements you want to insert=\n");
     scanf("%d",&n);
    printf("\n enter the elements in the ascending order= \n");
       for(i=0;i<n;i++)
            scanf("%d",&a[i]);
             printf("the array elements are = \n");
       for(i=0;i<n;i++)
              printf("%5d",a[i]);
              printf("\n which elements you want to search=");
              scanf("%d",&key);
                binary(a,n,key);
              getch();
  }

  void binary(int a[],int n,int key)
      {
            int low,high,mid;
            int flag=1;
            low=0;
               high=n-1;
                 while(low<=high)
                     {
                         mid=(low+high)/2;
                         if(key<a[mid])

                         high=mid-1;
                           else
                                if(key>a[mid])
                                  low=mid+1;
                                        else
                                           if(key==a[mid])
                                           {
                        printf("\n search successful \n");
    printf("\n elements %d found at location %d \n",key,mid+1);
    flag=0;
   break;
   }
  }
     if(flag)
       printf("\n unsuccessful search %d not found",key);
  }
```

OUTPUT :

```
 enter how many elements you want to insert=
5

 enter the elements in the ascending order=
5
7
8
9
11
the array elements are =
   5    7    8    9   11
 which elements you want to search=11

 search successful

 elements 11 found at location 5
```

```
 enter how many elements you want to insert=
5

 enter the elements in the ascending order=
5
7
8
9
11
the array elements are =
   5    7    8    9   11
 which elements you want to search=3

 unsuccessful search 3 not found
```

Viva Questions

1.Define Search.

2.Compare linear and binary searches.

3.which search is faster and why?

4. Give the time complexity of searching and sortngs

5. Give one numerical example for linear and binary searches.

Week8:

Write a program to implement the tree traversal methods.

Source code:

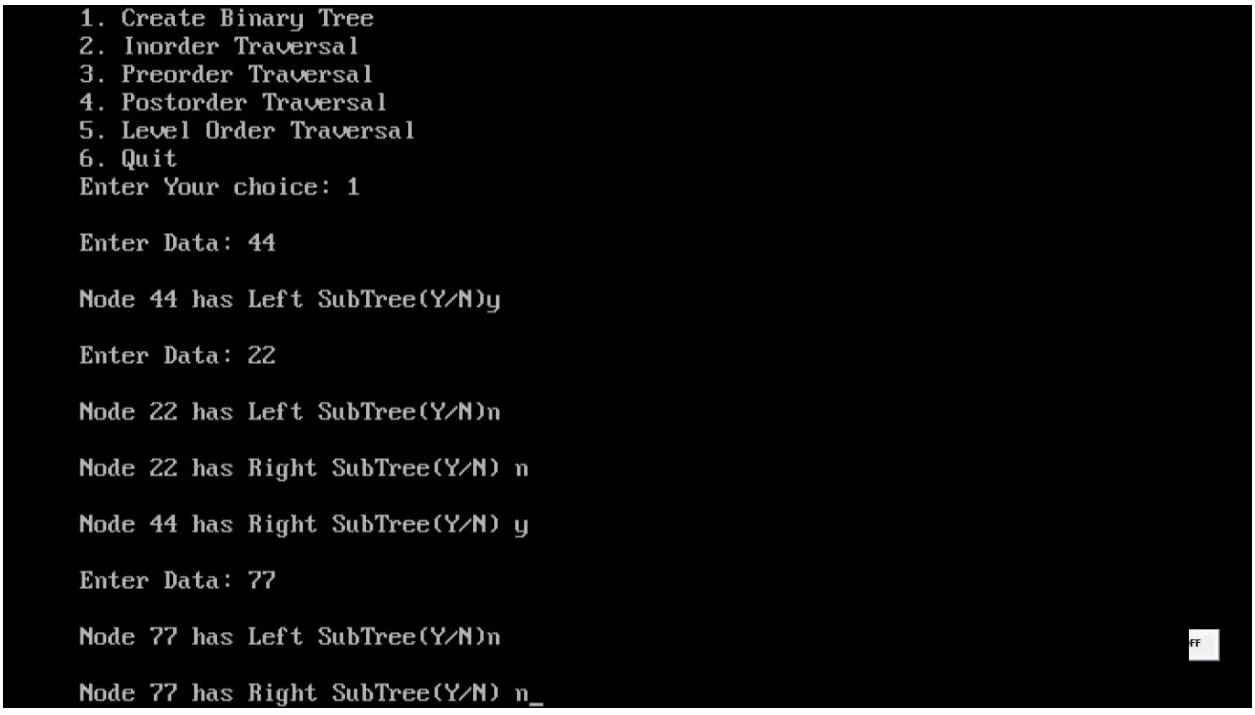
```
# include <stdio.h>
# include <stdlib.h>
  struct tree
   {
     struct tree* lchild;
     char data[10];
     struct tree* rchild;
    };

    typedef struct tree node;
    node *Q[50];
    int  node_ctr;

     node* getnode()
      {
        node *temp ;
         temp = (node*) malloc(sizeof(node));
         printf("\n Enter Data: ");
        fflush(stdin);
        scanf("%s",temp->data);
            temp->lchild = NULL;
            temp->rchild = NULL;
         return temp;
       }


   void create_binarytree(node *root)
    {
       char option;
        node_ctr = 1;
         if( root != NULL )
         {
         printf("\n Node %s has Left SubTree(Y/N)",root->data);
         fflush(stdin);
         scanf("%c",&option);
         if( option=='Y' || option == 'y')
          {
            root->lchild = getnode();
            node_ctr++;
            create_binarytree(root->lchild);
           }
         else
          {
         root->lchild = NULL;
         create_binarytree(root->lchild);
         }
         printf("\n Node %s has Right SubTree(Y/N) ",root->data);
         fflush(stdin);
```

```
            scanf("%c",&option);
            if( option=='Y' || option == 'y')
              {
               root->rchild = getnode();
                node_ctr++;
                create_binarytree(root->rchild);
               }
            else
             {
            root->rchild = NULL;
            create_binarytree(root->rchild);
            }
          }
      }

 void make_Queue(node *root,int parent)
  {
      if(root != NULL)
       {
          node_ctr++;
          Q[parent] = root;
          make_Queue(root->lchild,parent*2+1);
          make_Queue(root->rchild,parent*2+2);
        }
    }

void inorder(node *root)
 {
    if(root != NULL)
     {
        inorder(root->lchild);
        printf("%3s",root->data);
        inorder(root->rchild);
      }
   }



void  preorder(node *root)
  {
     if( root != NULL )
       {
        printf("%3s",root->data);
        preorder(root->lchild);
        preorder(root->rchild);
       }
   }

void postorder(node *root)
{
if( root != NULL )
{
postorder(root->lchild);
postorder(root->rchild);
```

```
printf("%3s", root->data);
}
}



void level_order(node *Q[],int ctr)
{
  int i;
  for( i = 0; i < ctr ; i++)
  {
   if( Q[i] != NULL )
         printf("%5s",Q[i]->data);
  }
 }

int menu()
{
  int ch;
  clrscr();
    printf("\n 1. Create Binary Tree ");
    printf("\n 2. Inorder Traversal ");
    printf("\n 3. Preorder Traversal ");
    printf("\n 4. Postorder Traversal ");
    printf("\n 5. Level Order Traversal");
    printf("\n 6. Quit ");
    printf("\n Enter Your choice: ");
    scanf("%d", &ch);
      return ch;
   }
void main()
{
        int i,ch;
        node *root = NULL;
        do
        {
        ch = menu();
        switch( ch)
        {
        case 1 :
                if( root == NULL )
                  {
                  root = getnode();
                  create_binarytree(root);
                  }
                else
                   {
                      printf("\n Tree is already Created ..");
                  }
                  break;
        case 2 :
                  printf("\n Inorder Traversal: ");
                  inorder(root);
                  break;
```

```
        case 3 :
                printf("\n Preorder Traversal: ");
                preorder(root);
                break;

        case 4 :
                printf("\n Postorder Traversal: ");
                postorder(root);
                break;
        case 5:
                printf("\n Level Order Traversal ..");
                make_Queue(root,0);
                level_order(Q,node_ctr);
                break;
        case 6 :
                exit(0);
      }
 getch();
  }while(1);
}
```

OUTPUT :



```
1. Create Binary Tree
2. Inorder Traversal
3. Preorder Traversal
4. Postorder Traversal
5. Level Order Traversal
6. Quit
Enter Your choice: 2

Inorder Traversal:  22 44 77
```

```
1. Create Binary Tree
2. Inorder Traversal
3. Preorder Traversal
4. Postorder Traversal
5. Level Order Traversal
6. Quit
Enter Your choice:
4

Postorder Traversal:  22 77 44
```

Viva Questions

1. Define binary tree
2. List the properties of BST.
3. What is complete binary tree?
4. Give one example for Full binary tree.
5. Give the node structure of a tree.

Week 9 : Viva questions

1. ***Define graph.***
2. ***List graph representations.***
3. ***What is directed graph?***
4. ***What is back edge, forward edge and cross edge?***
5. ***What is weighted graph?***

```
# include <conio.h>
# include <string.h>
char  prefix[50];
char infix[50];
char opstack[50]; /*
operator stack */
int j, top = 0;
void insert_beg(char ch)
{
int k;
if(j == 0)
prefix[0] = ch;
else
{
for(k = j + 1; k > 0; k--
)
prefix[k] = prefix[k - 1];
prefix[0] = ch;
}
j++;
}
int lesspriority(char op,
char op_at_stack)
{
int k;
int pv1; /* priority value
of op */
int pv2; /* priority value
of op_at_stack */
char operators[] = {'+',
'-', '*', '/', '%', '^', ')'};
int priority_value[] =
{0, 0, 1, 1, 2, 3, 4};
if(op_at_stack == ')' )
return 0;
for(k = 0; k < 6; k ++)
{
if(op == operators[k])
pv1 = priority_value[k];
}
for(k = 0; k < 6; k ++)
{
if( op_at_stack ==
operators[k] )
pv2 = priority_value[k];
}
if(pv1 < pv2)
return  1;
else
return 0;
}
void push(char op) /* op
– operator */
{
if(top == 0)
{
opstack[top] = op;
top++;
}
else
{
if(op != ')')
{
/* before pushing the
operator 'op' into the
```

```
stack check priority of
op
with top of operator
stack if less pop the
operator from stack then
push into
postfix string else push
op onto stack itself */
while(lesspriority(op,
opstack[top-1]) == 1
&& top > 0)
{
insert_beg(opstack[--
top]);
}
}
opstack[top] = op; /*
pushing onto stack */
top++;
}
}
void pop()
{
while(opstack[--top] !=
')') /* pop until ')'
comes; */
insert_beg(opstack[top]
);
}
        void main()
{
char  ch;
int l, i = 0;
clrscr();
printf("\n Enter
InfixExpression : ");
gets(infix);
l = strlen(infix);
while(l > 0)
{
ch = infix[--l];
switch(ch)
{
case ' ' : break;
case ')' :
case '+' :
case '-' :
case '*' :
case '/' :
case '^' :
case '%' :
push(ch); /* check
priority and push */
break;
case '(' :
pop();
break;
default :
insert_beg(ch);
}
}
while( top > 0 )
{
insert_beg( opstack[--
top]);
```

```
}
prefix[j] = '\0';
printf("\nInfix
Expression : %s ",
infix);
printf("\n Prefix
Expression : %s ",
prefix);
getch();

}
```

Program to evaluate a postfix expression

```
# include <conio.h>
# include <math.h>
# define MAX 20
int isoperator(char ch)
{
if(ch == '+' || ch == '-' || ch == '*' || ch == '/' || ch == '^')
return 1;
else
return 0;
}

void main(void)
{
char postfix[MAX];
int val;
char ch;
int i = 0, top = 0;
float val_stack[MAX], val1, val2, res;
clrscr();
printf("\n Enter a postfix expression: ");
scanf("%s", postfix);
while((ch = postfix[i]) != '\0')
{
if(isoperator(ch) == 1)
{
val2= val_stack[--top];
val1= val_stack[--top];
switch(ch)
{
case '+':
res = val1 + val2;
break;
case '-':
res = val1 - val2;
break;
case '*':
res = val1 * val2;
break;
case '/':
res = val1 / val2;
break;
case '^':
res = pow(val1, val2);
break;
}
val_stack[top] = res;
}
else
val_stack[top] = ch-48; /*convert character digit to integer digit */
top++;
i++;
}
printf("\n Values of %s is : %f ",postfix, val_stack[0] );
getch();
}
```

PROGRAM TO IMPLEMENT HEAPSORT

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

void adjust(int i, int n, int a[])
{
int  j, item;
j = 2 * i;
item = a[i];
while(j <= n)
{
if((j < n) && (a[j] < a[j+1]))
j++;
if(item >= a[j])
        break;
else
 {
   a[j/2] = a[j];
   j=2*j;
}
}
a[j/2] = item;
}

void heapify(int n, int a[])
{
    int  i;
    for(i = n/2; i > 0; i--)
    adjust(i, n, a);
 }

void heapsort(int n,int a[])
{
   int temp, i;
    heapify(n, a);
    for(i = n; i > 0; i--)
          {
          temp = a[i];
          a[i] = a[1];
          a[1]=temp;
          adjust(1, i - 1, a);
          }
}

void main()
{
   int i, n, a[20];
          clrscr();
    printf("\n How many element you want: ");
    scanf("%d",&n);
    printf("Enter %d elements: ", n);
    for (i=1; i<=n; i++)
        scanf("%d", &a[i]);
    heapsort(n,a);
  printf("\n The sorted elements are: \n");
      for (i=1; i<=n; i++)
          printf("%5d", a[i]);
   getch();
 }
```

Program to implement merge sort

```
#include<stdio.h>
#include<conio.h>
  void  main( )
   {
      int a[20],b[20],c[20],i,j,k,temp,n;

       printf(“enter the size “);
      scanf(“%d”,&n);
      printf(“\n enter elements into array A”);
            for(i=0;i<n;i++)
               scanf(“%d”,&a[i]);
     printf(“\n enter elements into array B”);
            for(i=0;i<n;i++)
               scanf(“%d”,&b[i]);
      for(i=0;i<=n-2;i++)
          {
           for(j=i+1;j<=n-1;j++)
              {
                     If(a[i]>a[j])
                     {
                       temp=a[i];
                      a[i]=a[j];
                      a[j]=temp;
                     }
              }
          }
     for(i=0;i<=n-2;i++)
          {
           for(j=i+1;j<=n-1;j++)
              {
                     If(b[i]>b[j])
                     {
                       temp=b[i];
                      b[i]=b[j];
                      b[j]=temp;
                     }
```

```
            }
          }
        for(1=j=k=0;i<(2*n);)
          {
            If(a[j]<=b[k])
                C[i++]=a[j++];
            else
                c[i++]=b[k++];
            if(j==n ||
                k==n)
                break;
          }
        for( ;j<n; )
          c[i++]=a[j++];
        for( ;k<n; )
          c[i++]=b[k++];
     printf(" sorted elements (using merge sort) are  \n");
      for(i=0;i<(2*n);i++)
          printf("%d" ,
    c[i]);getch();
}
```