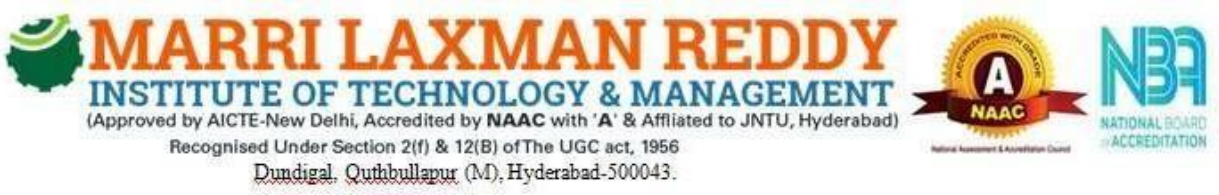


Data structures Lab Manual

Subject Code: CS505PC
Regulation: R19- JNTUH
Class:B.Tech II-I Semester

DEPARTMENT
OF
COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that this manual is a bonafide record of practical work in the **data structures** in **First Semester of II year B.Tech (CSE) programme** during the academic year **2019-20**. This book is prepared by **Mr.Sivaramakrishna (Asst.Professor)** Department of Computer Science and Engineering.

INDEX

PREFACE

This book “data structures” lab manual is intended to teach the design and analysis of basic data structures and their implementation in an object-oriented language. Readers of this book need only be familiar with the basic syntax of Java and similar languages. The “data structures Concepts” is increasingly becoming the default choice of the IT industry especially industries involved in software development at system level. Therefore, for proper development of “Object Oriented Programming ” skills among the students this practical manual has been prepared. The manual contains the exercise programs and their solution for easy & quick understanding of the students. We hope that this practical manual will be helpful for students of Computer Science & Engineering for understanding the subject from the point of view of applied aspects. There is always scope for improvement in the manual. We would appreciate to receive valuable suggestions from readers and users for future use.

By

Sivaramakrishna
K.Shalini,
sravani.

ACKNOWLEDGEMENT

It was really a good experience, working with "data structures **lab"**. First we would like to thank Mr.K.Abdul Basith, Assoc.Professor, HOD of Department of Computer Science and Engineering, Marri Laxman Reddy Institute of technology & Management for his concern and giving the technical support in preparing the document.

We are deeply indebted and gratefully acknowledge the constant support and valuable patronage of Dr.R.Kotaih, Director, Marri Laxman Reddy Institute of technology & Management for giving us this wonderful opportunity for preparing the data structures laboratory manual.

We express our hearty thanks to Dr.K.Venkateswara Reddy, Principal, Marri Laxman Reddy Institute of technology & Management, for timely corrections and scholarly guidance.

At last, but not the least I would like to thanks the entire CSE Department faculties those who had inspired and helped us to achieve our goal.

By

Sivaramakrishna

GENERAL INSTRUCTIONS

1. Students are instructed to come to Datasrutures laboratory on time. Late comers arenot entertained in the lab.

2. Students should be punctual to the lab. If not, the conducted experiments will not berepeated.

3. Students are expected to come prepared at home with the experiments which are going to beperformed.

4. Students are instructed to display their identity cards before entering into thelab.

5. Students are instructed not to bring mobile phones to the lab.

6. Any damage/loss of system parts like keyboard, mouse during the lab session, it is student's responsibility and penalty or fine will be collected from the student.

7. Students should update the records and lab observation books session wise. Before leaving the labthe student should get his lab observation book signed by the faculty.

8. Students should submit the lab records by the next lab to the concerned faculty members in thestaffroom for their correction and return.

9. Students should not move around the lab during the lab session.

10. If any emergency arises, the student should take the permission from faculty member concernedin written format.

11. The faculty members may suspend any student from the lab session on disciplinary grounds.

12. Never copy the output from other students. Write down your own outputs.

INSTITUTION VISION AND MISSION

VISION

To establish as an ideal academic institutions in the service of the nation, the world and the humanity by graduating talented engineers to be ethically strong, globally competent by conducting high quality research, developing breakthrough technologies, and disseminating and preserving technical knowledge.

MISSION

To fulfill the promised vision through the following strategic characteristics and aspirations:

- Contemporary and rigorous educational experiences that develop the engineers and managers.
- An atmosphere that facilitates personal commitment to the educational success of students in an environment that values diversity and community.
- Undergraduate programs that integrate global awareness, communication skills and team building.
- Education and Training that prepares students for interdisciplinary engineering research and advanced problem solving abilities.

DEPARTMENT VISION, MISSION , PROGRAMME EDUCATIONAL OBJECTIVES AND SPECIFIC OUTCOMES

VISION

The Computer science Engineering department strives to impart quality education by extracting the innovative skills of students and to face the challenges in latest technological advancements and to serve the society.

MISSION

Provide quality education and to motivate students towards professionalism

Address the advanced technologies in research and industrial issues

PROGRAMME EDUCATIONAL OBJECTIVES

The Programme Educational Objectives (PEOs) that are formulated for the computer science engineering programme are listed below;

PEO-I solving computer science engineering problems in different circumstances PEO-II Pursue higher education and research for professional development.

PEO-III Inculcate qualities of leadership for technology innovation and entrepreneurship.

PROGRAM SPECIFIC OUUTCOMES

PSO1. UNDERSTANDING: Graduates will have an ability to describe, analyze, and solve problems using mathematics and systematic problem-solving techniques.

PSO2. ANALYTICAL SKILLS: Graduates will have an ability to design a system, component, or process to meet desired needs within realistic constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability, and sustainability.

PSO3. BROADNESS: Graduates will have a broad education necessary to understand the impact of engineering solutions in a global, economic, and societal context.

PROGRAMME OUT COMES

a) An ability to apply knowledge of mathematics, science, and engineering

Graduates should transform knowledge of mathematics, Physics, chemistry, Engineering Mechanics, probability and statistics, and engineering drawing in solving a wide range of computer science engineering problems.

b) An ability to design, implement, evaluate a system and conduct experiments, as well as to analyze and interpret data

Graduates should show that they can make decisions regarding type, and number of data points to be collected, duration of the experiment to obtain intended results, and demonstrate an understanding of accuracy and precision of data

(c) **An ability to design, implement and evaluate a system, or process to meet desired needs** Graduates should be able to: identify the project goal; define the project; search for alternative possibilities; choose the best of the possible solutions; create a design drawing, design plan, or computer simulation; evaluate the design; and justify the final design in written and oral forms.

d) An ability to function effectively on multi-disciplinary teams

Graduates should show that they can participate effectively as team members with people who bring different skills, expertise, and perspectives to a project; and with people from different sub-disciplines of computer science engineering and interdisciplinary groups.

e) An ability to identify, formulate, analyse and solve engineering problems

Graduates should be able to describe the important components of a given problem, apply mathematical, engineering principles and to find the unknowns and arrive at appropriate and effective solutions.

f) An understanding of professional, ethical, legal, security and social responsibilities Graduates should be familiar with the applicable professional code of conduct for engineers. They should be able to apply the codes, where appropriate, to particular cases in which ethical issues arise. Graduates should also understand the importance of professionalism.

g) An ability to communicate effectively both in writing and orally

Computer science engineering graduates should have the ability to speak and write effectively in various domains like laboratory reports, technical reports, technical presentations, project reports etc.

h) The broad education necessary to analyse the impact of engineering solutions on a global and societal context

Graduates should get exposed to the interactions among science, technology, and social values, understand the influence of science and technology on computer scienceizations and how science and technology have been addressed for the betterment of humankind.

i) Recognition of the need for, and an ability to engage in continuing professional development and life-long learning

Graduates should show that they appreciate the need for further education and self improvement, understand the value of professional licensure the necessity of continuing professional developments, and the value of membership in appropriate professional organizations.

j) Knowledge of contemporary issues

Graduates should have knowledge and understand selected contemporary technical and social issues relevant to their field of study.

k) An ability to use the techniques, skills, and modern engineering tools necessary for engineering practice

Graduates should have ability to use practical methods readily and effectively in the performance of engineering analysis and design. Graduates should be able to select and use modern engineering tools used by practicing engineers, including computer software such as computer aided drawing (CAD)

l) An ability to apply design and development principles in the construction of software andhardware systems of varying complexity

Computer science Graduates should have ability to design and develop principles involved in construction of different structures like buildings, shopping complexes, roads, water structures and to analyse the stability of structures using different softwares like stadpro. Studs etc.

COURSE STRUCTURE, OBJECTIVES & OUTCOMES

COURSE STRUCTURE

Data structures lab will have a continuous evaluation during 7th semester for 25 sessional marks and 50 end semester examination marks.

Out of the 25 marks for internal evaluation, day-to-day work in the laboratory shall be evaluated for 15 marks and internal practical examination shall be evaluated for 10 marks conducted by the laboratory teacher concerned.

The end semester examination shall be conducted with an external examiner and internal examiner. The external examiner shall be appointed by the principal / Chief Controller of examinations

COURSE OBJECTIVE

- To write programs in java to solve problems using divide and conquer strategy.
- To write programs in java to solve problems using backtracking strategy.
- To write programs in java to solve problems using greedy and dynamic programming techniques.

COURSE OUTCOME

Ability to write programs in java to solve problems using algorithm design techniques such as Divide and Conquer, Greedy, Dynamic programming, and Backtracking.

III. Course files – Soft copies

HODs and faculty members are directed to submit the soft copies of the course files in the director's office/L.M.S at the earliest, other wise it will be treated as not submitted

EXPERIMENT-1

Aim: Write a program that uses functions to perform the following operations on single linked list.

i)Creation ii) Insertion iii)Deletion iv)Traversal

```
#include <stdio.h>
#include <malloc.h>
#include <stdlib.h>
struct node {
  int value;
  struct node *next;
};
void insert();
void display();
void delete();
int count();
typedef struct node DATA_NODE;
DATA_NODE *head_node, *first_node, *temp_node = 0, *prev_node, next_node;
int data;

int main() {
  int option = 0;

  printf("Singly Linked List Example - All Operations\n");

  while (option < 5) {

    printf("\nOptions\n");
    printf("1 : Insert into Linked List \n");
    printf("2 : Delete from Linked List \n");
    printf("3 : Display Linked List\n");
    printf("4 : Count Linked List\n");
    printf("Others : Exit()\n");
    printf("Enter your option:");
    scanf("%d", &option);
    switch (option) {
      case 1:
        insert();
        break;
      case 2:
```

```
        delete();
        break;
      case 3:
        display();
        break;
      case 4:
        count();
        break;
      default:
        break;
    }
  }

  return 0;
}

void insert() {
  printf("\nEnter Element for Insert Linked List : \n");
  scanf("%d", &data);

  temp_node = (DATA_NODE *) malloc(sizeof (DATA_NODE));

  temp_node->value = data;

  if (first_node == 0) {
    first_node = temp_node;
  } else {
    head_node->next = temp_node;
  }
  temp_node->next = 0;
  head_node = temp_node;
  fflush(stdin);
}

void delete() {
  int countvalue, pos, i = 0;
  countvalue = count();
  temp_node = first_node;
```

```
  printf("\nDisplay Linked List : \n");

  printf("\nEnter Position for Delete Element : \n");
  scanf("%d", &pos);

  if (pos > 0 && pos <= countvalue) {
    if (pos == 1) {
      temp_node = temp_node -> next;
      first_node = temp_node;
      printf("\nDeleted Successfully \n\n");
    } else {
      while (temp_node != 0) {
        if (i == (pos - 1)) {
          prev_node->next = temp_node->next;
          if(i == (countvalue - 1))
          {
                        head_node = prev_node;
               }
          printf("\nDeleted Successfully \n\n");
          break;
        } else {
          i++;
          prev_node = temp_node;
          temp_node = temp_node -> next;
        }
      }
    }
  } else
    printf("\nInvalid Position \n\n");
}

void display() {
  int count = 0;
  temp_node = first_node;
  printf("\nDisplay Linked List : \n");
  while (temp_node != 0) {
    printf("# %d # ", temp_node->value);
    count++;
```

```
    temp_node = temp_node -> next;
  }
  printf("\nNo Of Items In Linked List : %d\n", count);
}

int count() {
  int count = 0;
  temp_node = first_node;
  while (temp_node != 0) {
    count++;
    temp_node = temp_node -> next;
  }
  printf("\nNo Of Items In Linked List : %d\n", count);
  return count;
}
```

Sample Output:

```
Singly Linked List Example - All Operations

Options
1 : Insert into Linked List
2 : Delete from Linked List
3 : Display Linked List
4 : Count Linked List
Others :  Exit()
Enter your option:1
Enter Element for Insert Linked List :
100

Options
1 : Insert into Linked List
2 : Delete from Linked List
3 : Display Linked List
4 : Count Linked List
Others :  Exit()
Enter your option:1

Enter Element for Insert Linked List :
```

```
200

Options
1 : Insert into Linked List
2 : Delete from Linked List
3 : Display Linked List
4 : Count Linked List
Others : Exit()
Enter your option:1

Enter Element for Insert Linked List :
300

Options
1 : Insert into Linked List
2 : Delete from Linked List
3 : Display Linked List
4 : Count Linked List
Others : Exit()
Enter your option:1

Enter Element for Insert Linked List :
400

Options
1 : Insert into Linked List
2 : Delete from Linked List
3 : Display Linked List
4 : Count Linked List
Others : Exit()
Enter your option:1

Enter Element for Insert Linked List :
500

Options
1 : Insert into Linked List
2 : Delete from Linked List
```

```
3 : Display Linked List
4 : Count Linked List
Others :  Exit()
Enter your option:3

Display Linked List :
# 100 # # 200 # # 300 # # 400 # # 500 #
No Of Items In Linked List : 5

Options
1 : Insert into Linked List
2 : Delete from Linked List
3 : Display Linked List
4 : Count Linked List
Others : Exit()
Enter your option:4

No Of Items In Linked List : 5

Options
1 : Insert into Linked List
2 : Delete from Linked List
3 : Display Linked List
4 : Count Linked List
Others :  Exit()
Enter your option:2

No Of Items In Linked List : 5

Display Linked List :

Enter Position for Delete Element :
3

Deleted Successfully



Options
```

```
1 : Insert into Linked List
2 : Delete from Linked List
3 : Display Linked List
4 : Count Linked List
Others : Exit()
Enter your option:3

Display Linked List :
# 100 # # 200 # # 400 # # 500 #
No Of Items In Linked List : 4

Options
1 : Insert into Linked List
2 : Delete from Linked List
3 : Display Linked List
4 : Count Linked List
Others : Exit()
Enter your option:2

No Of Items In Linked List : 4

Display Linked List :

Enter Position for Delete Element :
6

Invalid Position



Options
1 : Insert into Linked List
2 : Delete from Linked List
3 : Display Linked List
4 : Count Linked List
Others : Exit()
Enter your option:
5
-------------------------
```

```
(program exited with code: 0)


Press any key to continue . . .
```

viva-questions

1) What is a linked list?
2) What type of memory allocation is referred for linked lists?
3) Mention what is traversal in linked lists?
4) Describe what is node in linked list?
5) Mention what is singly linked list?
6) What is the difference between linear array and linked list?
7) Mention what are the applications of linked lists?
8) What does the dummy header in linked list contain?
9) Mention the steps to insert data at the starting of a singly linked list?
10) What is the difference between singly and doubly linked lists?
11)What are the applications that use double linked lists?
12) Explain how to add an item to the beginning of the list?
13) Mention what is the biggest advantage of linked lists?
14) Mention how to delete first node from singly linked list?
15) Mention how to display singly linked list from first to last?
16) Mention how to insert a new node in linked list where free node will be available?
17) Mention for which header list, you will found the last node contains the null pointer?
18) A linear collection of data elements where the linear node is given by means of pointer is called?
19) Consider an implementation of unsorted singly linked list. Suppose it has its representation with a head pointer only.Given the representation, which of the following operation can be implemented in O(1) time?
20). In linked list each node contain minimum of two fields. One field is data field to store the data second field i ?
21) What would be the asymptotic time complexity to add a node at the end of singly linked list, if the pointer is initially pointing to the head of the list?
22) What would be the asymptotic time complexity to add an element in the linked list?
23) What would be the asymptotic time complexity to find an element in the linked list?
24) What would be the asymptotic time complexity to insert an element at the second position in the linked list?
25) What is a linked list?
26) How many pointers are required to implement a simple linked list?
27) How many types of linked lists are there?
28) How to represent a linked list node?
29) Describe the steps to insert data at the starting of a singly linked list?
30) How to insert a node at the end of linked list?
31) How to insert a node in random location in the list?
32) How to delete a node from linked list?
33) How to reverse a singly linked list?
34) Compare linked lists and dynamic arrays?
35) What is a circular linked list?
36) What is the difference between singly and doubly linked lists?
37) What are the applications that use linked lists?
38) How to remove loops in a linked list (or) what are fast and slow pointers used for?
39) What will you prefer to use a singly or a doubly linked lists for traversing through a list of elements?
40) What is a data structure linked list?

41) How a linked list is represented?
42) What are the different types of linked list?
43) What are the basic operations supported by linked lists?
44) What is data structure doubly linked list?
45) How to represent doubly linked list?
46) How do you calculate the sum of two linked list using recursion?
47) How do you reverse every alternate k nodes of a linked list ?
48) How do add two numbers represented using linked list?
49) What is a linked list?
50) How many pointers are required to implement a simple linked list?

2. Write a program that uses functions to perform the following operations on doubly linked list:

i) Creation ii) Insertion iii) Deletion iv) Traversal

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

struct node
{
    struct node *prev;
    int n;
    struct node *next;
}*h,*temp,*temp1,*temp2,*temp4;

void insert1();
void insert2();
void insert3();
void traversebeg();
void traverseend(int);
void sort();
void search();
void update();
void delete();

int count = 0;

void main()
{
    int ch;

    h = NULL;
    temp = temp1 = NULL;

    printf("\n 1 - Insert at beginning");
    printf("\n 2 - Insert at end");
    printf("\n 3 - Insert at position i");
    printf("\n 4 - Delete at i");
    printf("\n 5 - Display from beginning");
    printf("\n 6 - Display from end");
    printf("\n 7 - Search for element");
    printf("\n 8 - Sort the list");
    printf("\n 9 - Update an element");
    printf("\n 10 - Exit");
```

```
    while (1)
    {
        printf("\n Enter choice : ");
        scanf("%d", &ch);
        switch (ch)
        {
        case 1:
            insert1();
            break;
        case 2:
            insert2();
            break;
        case 3:
            insert3();
            break;
        case 4:
            delete();
            break;
        case 5:
            traversebeg();
            break;
        case 6:
            temp2 = h;
            if (temp2 == NULL)
                printf("\n Error : List empty to display ");
            else
            {
                printf("\n Reverse order of linked list is : ");
                traverseend(temp2->n);
            }
            break;
        case 7:
            search();
            break;
        case 8:
            sort();
            break;
        case 9:
            update();
            break;
        case 10:
            exit(0);
        default:
            printf("\n Wrong choice menu");
        }
    }
}

/* TO create an empty node */
void create()
{
    int data;

    temp =(struct node *)malloc(1*sizeof(struct node));
    temp->prev = NULL;
    temp->next = NULL;
    printf("\n Enter value to node : ");
    scanf("%d", &data);
```

```
    temp->n = data;
    count++;
}

/*  TO insert at beginning */
void insert1()
{
    if (h == NULL)
    {
        create();
        h  =  temp;
        temp1 = h;
    }
    else
    {
        create();
        temp->next = h;
        h->prev = temp;
        h = temp;
    }
}

/* To insert at end */
void insert2()
{
    if (h == NULL)
    {
        create();
        h  =  temp;
        temp1 = h;
    }
    else
    {
        create();
        temp1->next = temp;
        temp->prev = temp1;
        temp1 = temp;
    }
}

/* To insert at any position */
void insert3()
{
    int pos, i = 2;

    printf("\n Enter position to be inserted : ");
    scanf("%d", &pos);
    temp2 = h;

    if ((pos < 1) || (pos >= count + 1))
    {
        printf("\n Position out of range to insert");
        return;
    }
    if ((h == NULL) && (pos != 1))
    {
        printf("\n Empty list cannot insert other than 1st position");
        return;
```

```
    }
    if ((h == NULL) && (pos == 1))
    {
        create();
        h = temp;
        temp1 = h;
        return;
    }
    else
    {
        while (i < pos)
        {
            temp2 = temp2->next;
            i++;
        }
        create();
        temp->prev = temp2;
        temp->next = temp2->next;
        temp2->next->prev = temp;
        temp2->next = temp;
    }
}

/* To delete an element */
void delete()
{
    int i = 1, pos;

    printf("\n Enter position to be deleted : ");
    scanf("%d", &pos);
    temp2 = h;

    if ((pos < 1) || (pos >= count + 1))
    {
        printf("\n Error : Position out of range to delete");
        return;
    }
    if (h == NULL)
    {
        printf("\n Error : Empty list no elements to delete");
        return;
    }
    else
    {
        while (i < pos)
        {
            temp2 = temp2->next;
            i++;
        }
        if (i == 1)
        {
            if (temp2->next == NULL)
            {
                printf("Node deleted from list");
                free(temp2);
                temp2 = h = NULL;
                return;
            }
```

```
        }
        if (temp2->next == NULL)
        {
            temp2->prev->next = NULL;
            free(temp2);
            printf("Node deleted from list");
            return;
        }
        temp2->next->prev = temp2->prev;
        if (i != 1)
            temp2->prev->next = temp2->next;    /* Might not need this statement if i == 1 check */
        if (i == 1)
            h = temp2->next;
        printf("\n Node deleted");
        free(temp2);
    }
    count--;
}

/* Traverse from beginning */
void traversebeg()
{
    temp2 = h;

    if (temp2 == NULL)
    {
        printf("List empty to display \n");
        return;
    }
    printf("\n Linked list elements from begining : ");

    while (temp2->next != NULL)
    {
        printf(" %d ", temp2->n);
        temp2 = temp2->next;
    }
    printf(" %d ", temp2->n);
}

/* To traverse from end recursively */
void traverseend(int i)
{
    if (temp2 != NULL)
    {
        i = temp2->n;
        temp2 = temp2->next;
        traverseend(i);
        printf(" %d ", i);
    }
}

/* To search for an element in the list */
void search()
{
    int data, count = 0;
    temp2 = h;

    if (temp2 == NULL)
```

```
    {
        printf("\n Error : List empty to search for data");
        return;
    }
    printf("\n Enter value to search : ");
    scanf("%d", &data);
    while (temp2 != NULL)
    {
        if (temp2->n == data)
        {
            printf("\n Data found in %d position",count + 1);
            return;
        }
        else
             temp2 = temp2->next;
            count++;
    }
    printf("\n Error : %d not found in list", data);
}

/* To update a node value in the list */
void update()
{
    int data, data1;

    printf("\n Enter node data to be updated : ");
    scanf("%d", &data);
    printf("\n Enter new data : ");
    scanf("%d", &data1);
    temp2 = h;
    if (temp2 == NULL)
    {
        printf("\n Error : List empty no node to update");
        return;
    }
    while (temp2 != NULL)
    {
        if (temp2->n == data)
        {

            temp2->n = data1;
            traversebeg();
            return;
        }
        else
            temp2 = temp2->next;
    }

    printf("\n Error : %d not found in list to update", data);
}

/* To sort the linked list */
void sort()
{
    int i, j, x;

    temp2 = h;
    temp4 = h;
```

```
    if (temp2 == NULL)
    {
        printf("\n List empty to sort");
        return;
    }

    for (temp2 = h; temp2 != NULL; temp2 = temp2->next)
    {
        for (temp4 = temp2->next; temp4 != NULL; temp4 = temp4->next)
        {
            if (temp2->n > temp4->n)
            {
                x = temp2->n;
                temp2->n = temp4->n;
                temp4->n = x;
            }
        }
    }
    traversebeg();
}
```

Sample Output:

```
1 - Insert at beginning
2 - Insert at end
3 - Insert at position i
4 - Delete at i
5 - Display from beginning
6 - Display from end
7 - Search for element
8 - Sort the list
9 - Update an element
10 - Exit
Enter choice : 1

Enter value to node : 10

Enter choice : 2

Enter value to node : 50

Enter choice : 4

Enter position to be deleted : 1

Node deleted
Enter choice : 1

Enter value to node : 34

Enter choice : 3

Enter position to be inserted : 2

Enter value to node : 13

Enter choice : 4
```

```
Enter position to be deleted : 4

Error : Position out of range to delete
Enter choice : 1

Enter value to node : 15

Enter choice : 1

Enter value to node : 67

Enter choice : 3

Enter position to be inserted : 2

Enter value to node : 34

Enter choice : 4

Enter position to be deleted : 3

Node deleted
Enter choice : 7

Enter value to search : 15

Error : 15 not found in list
Enter choice : 8

Linked list elements from begining :  13  34  34  50  67
Enter choice : 9

Enter node data to be updated : 45

Enter new data : 89

Error : 45 not found in list to update
Enter choice : 9

Enter node data to be updated : 50

Enter new data : 90
Enter choice : 5

Linked list elements from begining :  13  34  34  90  67
Enter choice : 6

Reverse order of linked list is :  67  90  34  34  13
Enter choice : 7

Enter value to search : 90

Data found in 4 position
Enter choice : 8

Linked list elements from begining :  13  34  34  67  90
Enter choice : 7
```

```
Enter value to search : 90

Data found in 5 position
Enter choice : 9

Enter node data to be updated : 34

Enter new data : 56

Linked list elements from begining :  13  56  34  67  90
Enter choice : 10
```

viva-questions

1.What is data structure doubly linked list?
2. How to represent doubly linked list?
3. What are the basic operations supported by doubly linked list?
4.What is the advantage of linked list?
5. What are the applications of double linked list?
6. Write an algorithm to convert a binary tree into a doubly linked list?
7. How to find the length of a singly linked list?
8. How to remove duplicate nodes in an unsorted linked list?
9. Write code to print out the data stored in each node in a singly linked list?
10. Write a program to print a linked list in reverse order? e.g. print linked list from tail to head?
11. How to insert a node at the end of the list?
12. How to delete alternate nodes of a linked list?
13. What is the difference between an array and linked list?
14. Differentiate between singly and doubly linked list .
15. How to implement a linked list ?
16. How to insert a node at the beginning of the double linked list?
18. How do you traverse a linked list?
19. How do you find the sum of two linked list using stack?
20. How do you convert a sorted doubly linked list to a balanced binary search tree?
21. How do you calculate the sum of two linked list using recursion?
23. How do you reverse every alternate k nodes of a linked list?
24. How do add two numbers represented using linked list?
26. How many pointers are required to implement a simple linked list?
27. How many types of linked lists are there?
29. Describe the steps to insert data at the starting of a double linked list?
30. How to insert a node at the end of linked list?
31. How to insert a node in random location in the list?
32. How to delete a node from linked list?
33. How to reverse a double linked list?
34. Compare linked lists and dynamic arrays.
35. What is a circular double linked list?
36. What is the difference between singly and doubly linked lists?
37. What are the applications that uses double linked lists?
38. How to remove loops in a linked list?
39. What will you prefer to use a singly or a doubly linked lists for traversing through a list of elements?
40 .What is a data structure linked list?
41. What are the different types of linked lists?

42. What are the basic operations supported by linked lists?
43. What is data structure doubly linked list?
44. How to represent doubly linked list?
45. How do you calculate the sum of two linked list using recursion ?
46. How do you reverse every alternate k nodes of a linked list ?
47. Define double linked list?
48. What is a two linked list?
49. How many pointers are required to implement a double linked list?
50. Give an example for double linked list.

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

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

struct node
{
    int data;
    struct node *link;
};

struct node *head = NULL, *x, *y, *z;

void create();
void ins_at_beg();
void ins_at_pos();
void del_at_beg();
void del_at_pos();
void traverse();
void search();
void sort();
void update();
void rev_traverse(struct node *p);

void main()
{
    int ch;

    printf("\n 1.Creation \n 2.Insertion at beginning \n 3.Insertion at        remaining");
    printf("\n4.Deletion at beginning \n5.Deletion at remaining \n6.traverse");
    printf("\n7.Search\n8.sort\n9.update\n10.Exit\n");
    while (1)
    {
```

```
        printf("\n Enter your choice:");
        scanf("%d", &ch);
        switch(ch)
        {
        case 1:
            create();
            break;
        case 2:
            ins_at_beg();
            break;
        case 3:
            ins_at_pos();
            break;
        case 4:
            del_at_beg();
            break;
        case 5:
            del_at_pos();
            break;
        case 6:
            traverse();
            break;
        case 7:
            search();
            break;
        case 8:
            sort();
            break;
        case 9:
            update();
            break;
        case 10:
            rev_traverse(head);
            break;
        default:
            exit(0);
        }
    }
}

/*Function to create a new circular linked list*/
void create()
{
    int c;

    x = (struct node*)malloc(sizeof(struct node));
    printf("\n Enter the data:");
    scanf("%d", &x->data);
    x->link = x;
    head = x;
    printf("\n If you wish to continue press 1 otherwise 0:");
    scanf("%d", &c);
    while (c != 0)
    {
        y = (struct node*)malloc(sizeof(struct node));
```

```
            printf("\n Enter the data:");
            scanf("%d", &y->data);
            x->link = y;
            y->link = head;
            x = y;
            printf("\n If you wish to continue press 1 otherwise 0:");
            scanf("%d", &c);
        }
    }

    /*Function to insert an element at the begining of the list*/

    void ins_at_beg()
    {
        x = head;
        y = (struct node*)malloc(sizeof(struct node));
        printf("\n Enter the data:");
        scanf("%d", &y->data);
        while (x->link != head)
        {
            x = x->link;
        }
        x->link = y;
        y->link = head;
        head = y;
    }

    /*Function to insert an element at any position the list*/

    void ins_at_pos()
    {
        struct node *ptr;
        int c = 1, pos, count = 1;

        y = (struct node*)malloc(sizeof(struct node));
        if (head == NULL)
        {
            printf("cannot enter an element at this place");
        }
        printf("\n Enter the data:");
        scanf("%d", &y->data);
        printf("\n Enter the position to be inserted:");
        scanf("%d", &pos);
        x = head;
        ptr = head;
        while (ptr->link != head)
        {
            count++;
            ptr = ptr->link;
        }
        count++;
        if (pos > count)
        {
            printf("OUT OF BOUND");
            return;
```

```
    }
    while (c < pos)
    {
        z = x;
        x = x->link;
        c++;
    }
    y->link = x;
    z->link = y;
}

/*Function to delete an element at any begining of the list*/

void del_at_beg()
{
    if (head == NULL)
      printf("\n List is empty");
    else
    {
        x = head;
        y = head;
        while (x->link !=  head)
        {
            x = x->link;
        }
        head = y->link;
        x->link = head;
        free(y);
    }
}

/*Function to delete an element at any position the list*/

void del_at_pos()
{
    if (head == NULL)
        printf("\n List is empty");
    else
    {
        int c = 1, pos;
        printf("\n Enter the position to be deleted:");
        scanf("%d", &pos);
        x =  head;
        while (c < pos)
        {
            y = x;
            x = x->link;
            c++;
        }
        y->link = x->link;
        free(x);
    }
}

/*Function to display the elements in the list*/
```

```
void traverse()
{
    if (head == NULL)
        printf("\n List is empty");
    else
    {
        x = head;
        while (x->link !=  head)
        {
            printf("%d->", x->data);
            x = x->link;
        }
        printf("%d", x->data);
    }
}

/*Function to search an element in the list*/

void search()
{
    int search_val, count = 0, flag = 0;
    printf("\nenter the element to search\n");
    scanf("%d", &search_val);
    if (head == NULL)
        printf("\nList is empty nothing to search");
    else
    {
        x = head;
        while (x->link !=  head)
        {
            if (x->data == search_val)
            {
                printf("\nthe element is found at %d", count);
                flag = 1;
                break;
            }
            count++;
            x = x->link;
        }
        if (x->data == search_val)
        {
            printf("element found at postion %d", count);
        }
        if (flag == 0)
        {
            printf("\nelement not found");
        }

    }
}

/*Function to sort the list in ascending order*/

void sort()
```

```
{
    struct node *ptr, *nxt;
    int temp;

    if (head == NULL)
    {
        printf("empty linkedlist");
    }
    else
    {
        ptr = head;
        while (ptr->link !=  head)
        {
            nxt = ptr->link;
            while (nxt !=  head)
            {
                if (nxt !=  head)
                {
                    if (ptr->data > nxt->data)
                    {
                        temp = ptr->data;
                        ptr->data = nxt->data;
                        nxt->data = temp;
                    }
                }
                else
           {
               break;
           }
           nxt = nxt->link;
       }
       ptr = ptr->link;
         }
    }
}

/*Function to update an element at any position the list*/
void update()
{
    struct node *ptr;
    int search_val;
    int replace_val;
    int flag = 0;

    if (head == NULL)
    {
        printf("\n empty list");
    }
    else
    {
        printf("enter the value to be edited\n");
        scanf("%d", &search_val);
        fflush(stdin);
        printf("enter the value to be replace\n");
        scanf("%d", &replace_val);
```

```
            ptr = head;
            while (ptr->link !=  head)
            {
                if (ptr->data == search_val)
                {
                    ptr->data = replace_val;
                    flag = 1;
                    break;
                }
                ptr = ptr->link;
            }
            if (ptr->data == search_val)
            {
                ptr->data = replace_val;
                flag = 1;
            }
            if (flag == 1)
            {
                printf("\nUPdate sucessful");
            }
            else
            {
                printf("\n update not successful");
            }
        }
    }

    /*Function to display the elements of the list in reverse order*/

    void rev_traverse(struct node *p)
    {
        int i = 0;

        if (head == NULL)
        {
            printf("empty linked list");
        }
        else
        {
            if (p->link !=  head)
            {
                i = p->data;
                rev_traverse(p->link);
                printf(" %d", i);
            }
            if (p->link == head)
            {
                printf(" %d", p->data);
            }
        }
    }
```

```
$ cc circular_singly_ll.c
$ a.out
1.Creation
```

```
2.Insertion at beginning
3.Insertion at remaining
4.Deletion at beginning
5.Deletion at remaining
6.traverse
7.Search
8.sort
9.update
10.Exit
Enter your choice:6
List is  empty
Enter your choice:5
List is  empty
Enter your choice:9
empty list
Enter your choice:7
enter the element to search
12
List is empty nothing to search
Enter your choice:1
Enter the data:10
If you wish to continue press 1 otherwise 0:0
Enter your choice:3
Enter the data:20
Enter the position to be inserted:5
OUT OF BOUND
Enter your choice:2
Enter the data:12
Enter your choice:6
12->10
Enter your choice:3
Enter the data:13
Enter the position to be inserted:3
Enter your choice:3
Enter the data:14
Enter the position to be inserted:4
Enter your choice:6
12->10->13->14
Enter your choice:3
Enter the data:24
Enter the position to be inserted:4
Enter your choice:6
12->10->13->24->14
Enter your choice:3
Enter the data:10
Enter the position to be inserted:100
OUT OF BOUND
Enter your choice:4
Enter your choice:6
10->13->24->14
Enter your choice:5
Enter the position to be deleted:4
Enter your choice:6
10->13->24
Enter your choice:5
Enter the position to be deleted:2
Enter your choice:6
10->24
```

```
Enter your choice:9
enter the value to be edited
23
enter the value to be replace
24
update not successful
Enter your choice:9
enter the value to be edited
24
enter the value to be replace
26
UPdate sucessful
Enter your choice:6
10->26
Enter your choice:7
enter the element to search
26
element found at postion 1
element not found
Enter your choice:7
enter the element to search
27
element not found
Enter your choice:8
Enter your choice:6
10->26
Enter your choice:10
26 10
Enter your choice:11
```

viva-questions

1.What is data structure circular linked list?
2.What are the basic operations supported by the circular linked list?
3.What are the basic operations supported by doubly circular linked list?
4. Write a program to convert a binary tree into a circular linked list?
5. How to remove duplicate nodes in an unsorted linked list?
6. How to find the length of a circular singly linked list?
7. Write code to print out the data stored in each node in a circular linked list?
8. Write a program to print a linked list in reverse order?
9. How to find the kith node from the end in a circular linked list?
10. Write a program to print a linked list in reverse order?
11. How to insert a node at the end of the list?
12. How to delete alternate nodes of a linked list?
13. What is the difference between an array and linked list?
14. Differentiate between circular singly and circular doubly linked list?
16. How to insert a node at the beginning of the list?
17. How to insert a node at the end of the list?
18. How do you traverse a linked list?
19. How do you find the sum of two linked list using stack?
20. How do you convert a sorted doubly linked list to a balanced binary search tree?
21. How do you calculate the sum of two linked list using recursion?
23. How do you reverse every alternate k nodes of a linked list?

24. How do add two numbers represented using linked list?
25. What is a linked list?
26. How many pointers are required to implement a simple linked list?
27. How many types of linked lists are there?

28. How to represent a linked list node?
29. Describe the steps to insert data at the starting of a singly linked list?
30. How to insert a node at the end of linked list?
31. How to insert a node in random location in the list?
32. How to delete a node from linked list?
33. How to reverse a circular linked list?
34. Compare linked lists and dynamic arrays.
35. What is a circular linked list?
36. What is the difference between singly and circular linked lists?
37. What are the applications that use linked lists?
38. What are fast and slow pointers used for?
39. What will you prefer to use a singly or a doubly linked lists for traversing through a list of elements?
40. What is a data structure circular linked list?
41. How is a circular linked list is represented?
43. What are the basic operations supported by Linked Lists?
44 . What is data structure doubly linked list?
45 . What are the basic operations supported by doubly linked list?
46. How do you calculate the sum of two linked list using recursion?
47. How do you reverse every alternate k nodes of a linked list?
48. How do add two numbers represented using linked list?
49. What is a linked list?
50. How many pointers are required to implement a circular linked list?

4. Write a program that implement stack (its operations) using:

i) Arrays ii) Pointers

i) Arrays:

```
#include<stdio.h>
#define MAX  5
int top = -1;
int stack_arr[MAX];

/*Begin of push*/
void push()
{
        int pushed_item;
        if(top == (MAX-1))
                printf("Stack Overflow\n");
        else
        {
                printf("Enter the item to be pushed in stack : ");
                scanf("%d",&pushed_item);
                top=top+1;
                stack_arr[top] = pushed_item;
        }
}
/*End of push*/

/*Begin of pop*/
void pop()
{
        if(top == -1)
                printf("Stack Underflow\n");
        else
        {
                printf("Popped element is : %d\n",stack_arr[top]);
                top=top-1;
        }
}
/*End of pop*/

/*Begin of display*/
void display()
{
        int i;
        if(top == -1)
                printf("Stack is empty\n");
        else
        {
                printf("Stack elements :\n");
                for(i = top; i >=0; i--)
                        printf("%d\n", stack_arr[i] );
        }
}
/*End of display*/

/*Begin of main*/
main()
{
        int choice;

        do{
```

```
                printf("1.Push\n");
                printf("2.Pop\n");
                printf("3.Display\n");
                printf("4.Quit\n");
                printf("Enter your choice : ");
                scanf("%d",&choice);
                switch(choice)
                {
                 case 1 :
                        push();
                        break;
                 case 2:
                        pop();
                        break;
                 case 3:
                        display();
                        break;
                 case 4:
                          break;
                 default:
                        printf("Wrong choice\n");
                }}while(choice!=4);
}
/*End of main*/
```

Sample Output:

```
-------------------------------------------------------
    STACK IMPLEMENTATION PROGRAM
-------------------------------------------------------
1. Push
2. Pop
3. Size
4. Exit
-------------------------------------------------------
Enter your choice: 2
Stack is empty.


-------------------------------------------------------
    STACK IMPLEMENTATION PROGRAM
-------------------------------------------------------
1. Push
2. Pop
3. Size
4. Exit
-------------------------------------------------------
Enter your choice: 1
Enter data to push into stack: 10
Data pushed to stack.


-------------------------------------------------------
    STACK IMPLEMENTATION PROGRAM
-------------------------------------------------------
1. Push
2. Pop
3. Size
4. Exit
```

```
------------------------------------------------------------
Enter your choice: 1
Enter data to push into stack: 20
Data pushed to stack.


------------------------------------------------------------
    STACK IMPLEMENTATION PROGRAM
------------------------------------------------------------
1. Push
2. Pop
3. Size
4. Exit
------------------------------------------------------------
Enter your choice: 1
Enter data to push into stack: 30
Data pushed to stack.


------------------------------------------------------------
    STACK IMPLEMENTATION PROGRAM
------------------------------------------------------------
1. Push
2. Pop
3. Size
4. Exit
------------------------------------------------------------
Enter your choice: 3
Stack size: 3


------------------------------------------------------------
    STACK IMPLEMENTATION PROGRAM
------------------------------------------------------------
1. Push
2. Pop
3. Size
4. Exit
------------------------------------------------------------
Enter your choice: 2
Data => 30


------------------------------------------------------------
    STACK IMPLEMENTATION PROGRAM
------------------------------------------------------------
1. Push
2. Pop
3. Size
4. Exit
------------------------------------------------------------
Enter your choice: 2
Data => 20


------------------------------------------------------------
    STACK IMPLEMENTATION PROGRAM
------------------------------------------------------------
1. Push
2. Pop
3. Size
```

```
4. Exit
------------------------------------------------------------
Enter your choice: 2
Data => 10


------------------------------------------------------------
     STACK IMPLEMENTATION PROGRAM
------------------------------------------------------------
1. Push
2. Pop
3. Size
4. Exit
------------------------------------------------------------
Enter your choice: 2
Stack is empty.


------------------------------------------------------------
     STACK IMPLEMENTATION PROGRAM
------------------------------------------------------------
1. Push
2. Pop
3. Size
4. Exit
------------------------------------------------------------
Enter your choice: 4
Exiting from app.
```

ii) Pointers:

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

struct node
{
    int info;
    struct node *ptr;
}*top,*top1,*temp;

int topelement();
void push(int data);
void pop();
void empty();
void display();
void destroy();
void stack_count();
void create();

int count = 0;

void main()
{
    int no, ch, e;

    printf("\n 1 - Push");
    printf("\n 2 - Pop");
    printf("\n 3 - Top");
    printf("\n 4 - Empty");
    printf("\n 5 - Exit");
    printf("\n 6 - Dipslay");
    printf("\n 7 - Stack Count");
    printf("\n 8 - Destroy stack");

    create();

    while (1)
    {
        printf("\n Enter choice : ");
        scanf("%d", &ch);

        switch (ch)
        {
        case 1:
            printf("Enter data : ");
            scanf("%d", &no);
            push(no);
            break;
        case 2:
            pop();
            break;
        case 3:
            if (top == NULL)
                printf("No elements in stack");
            else
            {
                e = topelement();
                printf("\n Top element : %d", e);
```

```
            }
            break;
        case 4:
            empty();
            break;
        case 5:
            exit(0);
        case 6:
            display();
            break;
        case 7:
            stack_count();
            break;
        case 8:
            destroy();
            break;
        default :
            printf(" Wrong choice, Please enter correct choice  ");
            break;
        }
    }
}

/* Create empty stack */
void create()
{
    top = NULL;
}

/* Count stack elements */
void stack_count()
{
    printf("\n No. of elements in stack : %d", count);
}

/* Push data into stack */
void push(int data)
{
    if (top == NULL)
    {
        top =(struct node *)malloc(1*sizeof(struct node));
        top->ptr = NULL;
        top->info = data;
    }
    else
    {
        temp =(struct node *)malloc(1*sizeof(struct node));
        temp->ptr = top;
        temp->info = data;
        top = temp;
    }
    count++;
}

/* Display stack elements */
void display()
{
    top1 = top;
```

```
    if (top1 == NULL)
    {
        printf("Stack is empty");
        return;
    }

    while (top1 != NULL)
    {
        printf("%d ", top1->info);
        top1 = top1->ptr;
    }
 }

/* Pop Operation on stack */
void pop()
{
    top1 = top;

    if (top1 == NULL)
    {
        printf("\n Error : Trying to pop from empty stack");
        return;
    }
    else
        top1 = top1->ptr;
    printf("\n Popped value : %d", top->info);
    free(top);
    top = top1;
    count--;
}

/* Return top element */
int topelement()
{
    return(top->info);
}

/* Check if stack is empty or not */
void empty()
{
    if (top == NULL)
        printf("\n Stack is empty");
    else
        printf("\n Stack is not empty with %d elements", count);
}

/* Destroy entire stack */
void destroy()
{
    top1 = top;

    while (top1 != NULL)
    {
        top1 = top->ptr;
        free(top);
        top = top1;
        top1 = top1->ptr;
```

```
    }
    free(top1);
    top = NULL;

    printf("\n All stack elements destroyed");
    count = 0;
}
```

Sample Output:

```
1 - Push
2 - Pop
3 - Top
4 - Empty
5 - Exit
6 - Dipslay
7 - Stack Count
8 - Destroy stack
Enter choice : 1
Enter data : 56

Enter choice : 1
Enter data : 80

Enter choice : 2

Popped value : 80
Enter choice : 3

Top element : 56
Enter choice : 1
Enter data : 78

Enter choice : 1
Enter data : 90

Enter choice : 6
90 78 56
Enter choice : 7

No. of elements in stack : 3
Enter choice : 8

All stack elements destroyed
Enter choice : 4

Stack  is  empty
Enter choice : 5
```

viva-questions

1) What is data structure?
3) When is a binary search best applied?

4) What is a linked list?
5) How do you reference all the elements in a one-dimension array?
6) In what areas do data structures are applied?
7) What is LIFO?
8) What is a queue?
9) What are binary trees?
10) Which data structures are applied when dealing with a recursive function?
11) What is a stack?
12) Define binary search tree.
13) What are multidimensional arrays?
14) Are linked lists considered linear or non-linear data structures?
15) How does dynamic memory allocation help in managing data?
16) What is FIFO?
17) What is an ordered list?
18) What is merge sort?
19) Differentiate NULL and VOID.
20) What is the primary advantage of a linked list?
22) What is a linear search?
23) How does variable declaration affect memory allocation?
24) What is the advantage of the heap over a stack?
25) What is a postfix expression?
26) What is data abstraction?

27) How do you insert a new item in a binary search tree?
28) How does a selection sort work for an array?
29) How do signed and unsigned numbers affect memory?
30) What is the minimum number of nodes that a binary tree can have?
31) What are dynamic data structures?
32) In what data structures are pointers applied?
33) Do all declaration statements result in a fixed reservation in memory?
34) What are arrays?
35) What is the minimum number of queues needed when implementing a priority queue?
36) Which sorting algorithm is considered the fastest?
37) Differentiate stack from array.
38) Give a basic algorithm for searching a binary search tree.
39) What is a dequeue?
40) What is a bubble sort and how do you perform it?
41) What are the parts of a linked list?
42) How does selection sort work?
43) What is a graph?
44) Differentiate linear from a nonlinear data structure.
45) What is an AVL tree?
46) What are doubly linked lists?
47) What is Huffman's algorithm?
48) What is Fibonacci search?
49) Briefly explain recursive algorithm.
50) How do you search for a target key in a linked list?

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

i) Arrays

```
#include <stdio.h>

#define MAX 50

void  insert();
void  delete();
void display();
int queue_array[MAX];
int rear = - 1;
int front = - 1;
main()
{
    int choice;
    while (1)
    {
        printf("1.Insert element to queue \n");
        printf("2.Delete element from queue \n");
        printf("3.Display all elements of queue \n");
        printf("4.Quit \n");
        printf("Enter your choice : ");
        scanf("%d", &choice);
        switch (choice)
        {
            case 1:
            insert();
            break;
            case 2:
            delete();
            break;
            case 3:
            display();
            break;
            case 4:
            exit(1);
            default:
            printf("Wrong choice \n");
        } /* End of switch */
    } /* End of while */
} /* End of main() */

void insert()
{
    int add_item;
    if (rear == MAX - 1)
    printf("Queue Overflow \n");
    else
    {
        if (front == - 1)
        /*If queue is initially empty */
```

```
        front = 0;
        printf("Inset the element in queue : ");
        scanf("%d", &add_item);
        rear = rear + 1;
        queue_array[rear] = add_item;
    }
} /* End of insert() */

void delete()
{
    if (front == - 1 || front > rear)
    {
        printf("Queue Underflow \n");
        return ;
    }
    else
    {
        printf("Element deleted from queue is : %d\n", queue_array[front]);
        front = front + 1;
    }
} /* End of delete() */

void display()
{
    int i;
    if (front == - 1)
        printf("Queue is empty \n");
    else
    {
        printf("Queue is : \n");
        for (i = front; i <= rear; i++)
            printf("%d ", queue_array[i]);
        printf("\n");
    }
} /* End of display() */

Sample Output:
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 1
Inset the element in queue : 10
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 1
Inset the element in queue : 15
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 1
Inset the element in queue : 20
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 1
```

```
Inset the element in queue : 30
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 2
Element deleted from queue is : 10
1.Insert element to  queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 3
Queue is :
15 20 30
1.Insert element to queue
2.Delete element from queue
3.Display all elements of queue
4.Quit
Enter your choice : 4
```

ii) Pointers:

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

struct node
{
    int info;
    struct node *ptr;
}*front,*rear,*temp,*front1;

int frontelement();
void enq(int data);
void deq();
void empty();
void display();
void create();
void queuesize();

int count = 0;

void main()
{
    int no, ch, e;

    printf("\n 1 - Enque");
    printf("\n 2 - Deque");
    printf("\n 3 - Front element");
    printf("\n 4 - Empty");
    printf("\n 5 - Exit");
    printf("\n 6 - Display");
    printf("\n 7 - Queue size");
    create();
    while (1)
    {
        printf("\n Enter choice : ");
        scanf("%d", &ch);
        switch (ch)
```

```
        {
        case 1:
            printf("Enter data : ");
            scanf("%d", &no);
            enq(no);
            break;
        case 2:
            deq();
            break;
        case 3:
            e = frontelement();
            if (e != 0)
                printf("Front element : %d", e);
            else
                printf("\n No front element in Queue as queue is empty");
            break;
        case 4:
            empty();
            break;
        case 5:
            exit(0);
        case 6:
            display();
            break;
        case 7:
            queuesize();
            break;
        default:
            printf("Wrong choice, Please enter correct choice  ");
            break;
        }
    }
}

/* Create an empty queue */
void create()
{
    front = rear = NULL;
}

/* Returns queue size */
void queuesize()
{
    printf("\n Queue size : %d", count);
}

/* Enqueing the queue */
void enq(int data)
{
    if (rear == NULL)
    {
        rear = (struct node *)malloc(1*sizeof(struct node));
        rear->ptr = NULL;
        rear->info = data;
        front = rear;
    }
    else
    {
        temp=(struct node *)malloc(1*sizeof(struct node));
        rear->ptr = temp;
        temp->info = data;
```

```
        temp->ptr = NULL;

        rear = temp;
    }
    count++;
}

/* Displaying the queue elements */
void display()
{
    front1 = front;

    if ((front1 == NULL) && (rear == NULL))
    {
        printf("Queue is empty");
        return;
    }
    while (front1 != rear)
    {
        printf("%d ", front1->info);
        front1 = front1->ptr;
    }
    if (front1 == rear)
        printf("%d", front1->info);
}

/* Dequeing the queue */
void deq()
{
    front1 = front;

    if (front1 == NULL)
    {
        printf("\n Error: Trying to display elements from empty queue");
        return;
    }
    else
        if (front1->ptr != NULL)
        {
            front1 = front1->ptr;
            printf("\n Dequed value : %d", front->info);
            free(front);
            front = front1;
        }
        else
        {
            printf("\n Dequed value : %d", front->info);
            free(front);
            front = NULL;
            rear = NULL;
        }
        count--;
}

/* Returns the front element of queue */
int frontelement()
{
    if ((front != NULL) && (rear != NULL))
        return(front->info);
    else
        return 0;
```

```
}

/* Display if queue is empty or not */
void empty()
{
     if ((front == NULL) && (rear == NULL))
        printf("\n Queue empty");
    else
       printf("Queue not empty");
}


Sample output:

1 - Enque
2 - Deque
3 - Front element
4 - Empty
5 - Exit
6 - Display
7 - Queue size
Enter choice : 1
Enter data : 14

Enter choice : 1
Enter data : 85

Enter choice : 1
Enter data : 38

Enter choice : 3
Front element : 14
Enter choice : 6
14 85 38
Enter choice : 7

Queue size : 3
Enter choice : 2

Dequed value : 14
Enter choice : 6
85 38
Enter choice : 7

Queue size : 2
Enter choice : 4
Queue not empty
Enter choice : 5
```

viva-questions

1. What is linear data structure?
2. What are the minimum number of queues needed to implement the priority queue?
3. What are the notations used in Evaluation of Arithmetic Expressions using prefix and postfix forms?
4. List out few of the Application of tree data-structure?
5. What is the type of the algorithm used in solving the 8 Queens problem?
6. In RDBMS, what is the efficient data structure used in the internal storage representation?

7. What is a spanning Tree?
8. List out the areas in which data structures are applied extensively?
9. Translate infix expression into its equivalent post fix expression: (A-B)*(D/E)
10. What are priority queues?
11. What is a string?
12. What is Brute Force algorithm?
13. What are the limitations of arrays?
14. How can you overcome the limitations of arrays?
15. What is a linked list?
16. What is a node?
17. What does node consist of?
18. What is a queue ?
19. What are the types of Collision Resolution Techniques and the methods used in each of the type?
20. What are the methods available in storing sequential files ?
21. Mention some of the problem solving strategies?
22. What is divide and conquer method?
23. What is the need for the header?
24. Define leaf.
25. What are the applications of binary tree?
26. What are the different types of traversing?

27. Define pre-order traversal?
28. Define post-order traversal?
29. Define in -order traversal?
30. What is meant by sorting?
31. What's the major distinction in between storage structure and file structure and how?
32. Stack can be described as a pointer. Explain?
33. What do you mean by: Syntax Error, Logical Error, Run time Error?
34. What is mean by d-queue?
35. What is AVL tree?
36. What is binary tree?
37. What is the difference between a stack and a Queue?
38. What actions are performed when a function is called?
39. What is precision?
40. What do you mean by overflow and underflow?
41. Define sink node.
42. List the two kinds of data structure.
43. Define tree.
44. Define graph.
45. What is binary tree?
46. Define complete binary tree?
47. What is Binary Search Tree?
48. What is called collections?
49. What are the applications of Stack?
50. What are the applications of queue?

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

i) Bubble sort :

```
#include <stdio.h>

 int main()

{

  int array[100], n, c, d, swap;

  printf("Enter number of elements\n");

  scanf("%d", &n);

   printf("Enter %d integers\n", n);

  for (c = 0; c < n; c++)

    scanf("%d", &array[c]);

   for (c = 0 ; c < n - 1; c++)

  {

    for (d = 0 ; d < n - c - 1; d++)

    {

      if (array[d] > array[d+1]) /* For decreasing order use < */

      {

        swap       = array[d];
```

```
        array[d]   = array[d+1];
        array[d+1] = swap;
      }
    }
  }
   printf("Sorted list in ascending order:\n");
  for (c = 0; c < n; c++)
     printf("%d\n", array[c]);
   return 0;
}
```

Sample Output:

```
Enter no:of elements: 5

Enter 5 numbers : 345 3 20 35 333

Sorted array is : 3 20 35 333 345
```

ii) Selection sort:

```
#include <stdio.h>


int main()
{
  int array[100], n, c, d, position, swap;


  printf("Enter number of elements\n");
  scanf("%d", &n);


  printf("Enter %d integers\n", n);


  for (c = 0; c < n; c++)
    scanf("%d", &array[c]);


  for (c = 0; c < (n - 1); c++)
  {
    position = c;
```

```
    for (d = c + 1; d < n; d++)
    {
      if (array[position] > array[d])
        position = d;
    }
    if (position != c)
    {
      swap = array[c];
      array[c] = array[position];
      array[position] = swap;
    }
  }

  printf("Sorted list in ascending order:\n");

  for (c = 0; c < n; c++)
    printf("%d\n", array[c]);

  return 0;
}
```

Sample Output:

```
Enter no:of elements: 5
 Enter 5 numbers : 34 13 204 355 333
 Sorted array is : 13 34 204 333 355
```

iii) Insertion sort:

```
#include <stdio.h>

int main()
{
  int n, array[1000], c, d, t;

  printf("Enter number of elements\n");
  scanf("%d", &n);

  printf("Enter %d integers\n", n);

  for (c = 0; c < n; c++)
    scanf("%d", &array[c]);

  for (c = 1 ; c <= n - 1; c++) {
    d = c;
```

```
    while ( d > 0 && array[d-1] > array[d]) {
      t          = array[d];
      array[d]   = array[d-1];
      array[d-1] = t;

      d--;
    }
  }

  printf("Sorted list in ascending order:\n");

  for (c = 0; c <= n - 1; c++) {
    printf("%d\n", array[c]);
  }

  return 0;
}
```

```
Sample Output:
enter elements to be sorted:8 2 4 9 3 6 1
sorted elements:
 1 2 3 4 6 8 9
```

viva-questions

1. Define searching process?
2. How many types of searching are there?
3. Define linear search?
4. Why binary search method is more efficient then liner search?

5 .Efficiency of linear search ?

6 .What do you understand by the term "linear search is unsuccessful"?

7 .What is worse case?

8 .What is the drawback of linear search?

9 .During linear search, when the record is present in first position then how many comparisons are made ?

10. During linear search, when the record is present in last position then how many comparisons are made?

11.What is an array & how many types of arrays represented in memory?

12.How can be declared an array?

13. How can be insert an element in an array?

14 .How can be delete an element in an array?

15 .How many types of implementation of a two-dimensional array?

16 .What is array of pointers?

17 .What are limitations of array?

18 .Where the elements of the array are stored respectively in successive?

19 .How can merge two arrays?

20 .What is the operations of array?

21 .What are structures?

22 .What are arrays?

23 Is array of structures possible?
24 How structures are declared?
25 Why we use functions?
26 Name some library functions?
27 What are user defined functions?
28 Explain call by value?

29. When the value is passed in the function is called call by value?
30 .Explain call by reference?
31 .When the address of the value is passed is called call by reference.
32 .Why we used keyword „break"?
33 .When break is encountered inside any loop, control automatically passes to the first statement after the loop.
34 .What is stack?

35 .What are the operations performed on stack? 36

What is push operation?

37 .What is pop operation.

38 .How stacks are implemented?

39 .What are the applications of stack?

40 .What is recursion?

41 .What are the different types of stacks?

42 .Define “Top of stack”.

43 .Is stack is prinmitive or non primitive data structure ?.

44. What are the applications of Stack ?

45 .What are the Terms used in Stack ?

46 .Explain infix in Stack ?

47 .Explain Postfix in Stack ?

48 .Define operations on Stack ?

49 .Give postfix form for (A+B)*C/D

50 .Give prefix form for A/B^C+D

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

i) Linear search

```
#include <stdio.h>
#define MAX_LEN 10

void l_search_recursive(int l[],int num,int ele);
void l_search_nonrecursive(int l[],int num,int ele);
void read_list(int l[],int num);
void print_list(int l[],int num);

int main()
{
    int l[MAX_LEN], num, ele;
    int ch;

    printf("======================================================");
    printf("\n\t\t\tMENU");
    printf("\n======================================================");
    printf("\n[1] Linear Search using Recursion method");
    printf("\n[2] Linear Search using Non-Recursion method");
    printf("\n\nEnter your Choice:");
    scanf("%d",&ch);

    if(ch<=2 & ch>0)
```

```
    {
        printf("Enter the number of elements :");
        scanf("%d",&num);
        read_list(l,num);
        printf("\nElements present in the list are:\n\n");
        print_list(l,num);
        printf("\n\nElement you want to search:\n\n");
        scanf("%d",&ele);

        switch(ch)
        {
        case 1:
            printf("\n**Recursion method**\n");
            l_search_recursive(l,num,ele);
            break;

        case 2:
            printf("\n**Non-Recursion method**\n");
            l_search_nonrecursive(l,num,ele);
            break;
        }
    }
}

//end main

//Non-Recursive method

void l_search_nonrecursive(int l[],int num,int ele)
{
    int j, f=0;
    for(j=0; j<num; j++)
        if( l[j] == ele)
        {
            printf("\nThe element %d is present at position %d in list\n",ele,j);
            f=1;
            break;
        }
    if(f==0)
        printf("\nThe element is %d is not present in the list\n",ele);
}

//Recursive method

void l_search_recursive(int l[],int num,int ele)
{
    int f = 0;

    if( l[num] == ele)
    {
        printf("\nThe element %d is present at position %d in list\n",ele,num);
        f=1;
    }
    else
    {
        if((num==0) && (f==0))
        {
            printf("The element %d is not found.",ele);
        }
        else
        {
            l_search_nonrecursive(l,num-1,ele);
```

```
        }
    }
    getch();
}

void read_list(int l[],int num)
{
    int j;
    printf("\nEnter the elements:\n");
    for(j=0; j<num; j++)
        scanf("%d",&l[j]);
}

void print_list(int l[],int num)
{
    int j;
    for(j=0; j<num; j++)
        printf("%d\t",l[j]);
}
```

Sample Output:

```
=======================================================

                        MENU

=======================================================

[1] Linear Search using Recursion method

[2] Linear Search using Non-Recursion method

Enter your Choice:1

Enter the number of elements :5

Enter the elements:4

8

9

7

6

Elements present in the list are: 4       8     9     7     6

Element you want to search:9

**Recursion method**

The element 9 is present at position 2 in list
```

ii) Binary search:

```
#include <stdio.h>
#define MAX_LEN 10

/* Non-Recursive function*/
void b_search_nonrecursive(int l[],int num,int ele)
{
   int l1,i,j, flag = 0;
   l1 = 0;
   i = num-1;
   while(l1 <= i)
   {
      j = (l1+i)/2;
      if( l[j] == ele)
      {
    printf("\nThe element %d is present at position %d in list\n",ele,j);
         flag =1;
         break;
      }
      else
        if(l[j] < ele)
           l1 = j+1;
        else
           i = j-1;
   }
   if( flag == 0)
   printf("\nThe element %d is not present in the list\n",ele);
}

/* Recursive function*/
int b_search_recursive(int l[],int arrayStart,int arrayEnd,int a)
{
  int m,pos;
  if (arrayStart<=arrayEnd)
  {
    m=(arrayStart+arrayEnd)/2;
    if (l[m]==a)
      return m;
    else if (a<l[m])
      return b_search_recursive(l,arrayStart,m-1,a);
    else
      return b_search_recursive(l,m+1,arrayEnd,a);
   }
   return -1;
}

void read_list(int l[],int n)
{
   int i;
   printf("\nEnter the elements:\n");
   for(i=0;i<n;i++)
       scanf("%d",&l[i]);
}

void print_list(int l[],int n)
{
    int i;
   for(i=0;i<n;i++)
       printf("%d\t",l[i]);
```

```
}

/*main function*/
main()
{
   int l[MAX_LEN], num, ele,f,l1,a;
   int ch,pos;

   printf("=======================================================");
   printf("\n\t\t\tMENU");
   printf("\n=======================================================");
   printf("\n[1] Binary Search using Recursion method");
   printf("\n[2] Binary Search using Non-Recursion method");
   printf("\n\nEnter your Choice:");
   scanf("%d",&ch);

   if(ch<=2 & ch>0)
   {
     printf("\nEnter the number of elements : ");
     scanf("%d",&num);
     read_list(l,num);
     printf("\nElements present in the list are:\n\n");
     print_list(l,num);
     printf("\n\nEnter the element you want to search:\n\n");
     scanf("%d",&ele);


   switch(ch)
   {
     case 1:printf("\nRecursive method:\n");
        pos=b_search_recursive(l,0,num,ele);
        if(pos==-1)
        {
        printf("Element is not found");
        }
        else
        {
        printf("Element is found at %d position",pos);
        }
        //getch();
        break;

     case 2:printf("\nNon-Recursive method:\n");
        b_search_nonrecursive(l,num,ele);
        //getch();
        break;
     }
   }
}
```

Sample Output:

```
=======================================================
                        MENU
=======================================================
```

```
[1] Binary Search using Recursion method
[2] Binary Search using Non-Recursion method

Enter your Choice:1

Enter the number of elements : 5

Enter the elements:
12
22
32
42
52

Elements present in the list are:

12      22      32      42      52

Enter the element you want to search:

42

Recursive method:
Element is found at 3 position
```

viva-questions

1. Where is linear searching used?
2. What is the best case for linear search?
3. What is the worst case for linear search?
4. What is the best case and worst case complexity of ordered linear search?
5. What is the disadvantage of linear search?
6. Is there any difference in the speed of execution between linear serach(recursive) vs linear search(lterative)?
7. Is the space consumed by the linear search(recursive) and linear search(iterative) same?
8. What is the worst case runtime of linear search(recursive) algorithm?
9. Linear search(recursive) algorithm used in ?
10. The array is as follows: 1,2,3,6,8,10. At what time the element 6 is found? (By using linear search(recursive) algorithm)
11. The array is as follows: 1,2,3,6,8,10. Given that the number 17 is to be searched. At which call it tells that there's no such element? (By using linear search(recursive) algorithm)
12. What is the best case runtime of linear search(recursive) algorithm on an ordered set of elements?
13. Can linear search recursive algorithm and binary search recursive algorithm be performed on an unordered list?
14. What is the recurrence relation for the linear search recursive algorithm?
15. What is the advantage of recursive approach than an iterative approach?
16. Given an input arr = {2,5,7,99,899}; key = 899; What is the level of recursion?
17. Given an array arr = {45,77,89,90,94,99,100} and key = 99; what are the mid values(corresponding array elements) in the first and second levels of recursion?
18. What is the worst case complexity of binary search using recursion?
19. What is the average case time complexity of binary search using recursion?
20. What are the applications of binary search?

21. Binary Search can be categorized into which type of algorithm?
22. Given an array arr = {5,6,77,88,99} and key = 88; How many iterations are done until the element is found?
23. Given an array arr = {45,77,89,90,94,99,100} and key = 100; What are the mid values(corresponding array elements) generated in the first and second iterations?
24. What is the time complexity of binary search with iteration?
25. When is the uniform binary search an optimization over the usual binary search?
26. Given delta[4] is a global array and number of elements in the sorted array is 10, what are the values in the delta array?
27. What is the time complexity of uniform binary search?
28. Given, arr = {1,3,5,6,7,9,14,15,17,19} key = 17 and delta = {5,3,1,0}
    How many key comparisons are made?(exclude the comparison used to decide the left or right sub array)
29. How can Jump Search be improved?
30. Which search has efficiency of O(1)?
31. Time required to merge two sorted lists of size m and n, is?
32. Quick sort algorithm is an example of?
33. Which of the searching techniques do not require the data to be in sorted form?
34. Which of the sorting techniques has highest best-case runtime complexity?
35. Define adaptive sorting algorithm.
36. The worst case complexity of binary search matches with −?
37. The number of comparisons done by sequential search is ……
38. In ....., search start at the beginning of the list and check every element in the list.
39. State True or False.
    i) Binary search is used for searching in a sorted array.
    ii) The time complexity of binary search is O(logn).
40. In general, the binary search method needs no more than.....comparisons.
41. The Worst case occur in linear search algorithm when?
42. What is the average case in linear search algorithm?
43. Which is not the required condition for binary search algorithm?
44. Binary search algorithm cannot be applied to?
45. You have to sort a list L consisting of a sorted list followed by a few “random” elements. Which of the following sorting methods would be especially suitable for such a task?
46. A technique for direct search is
47. The searching technique that takes O (1) time to find a data is?
48. A sort which relatively passes through a list to exchange the first element with any element less than it and then repeats with a new first element is called?
49. The total number of companions required to merge 4 sorted files containing 15, 3, 9 and 8 records into a single sorted file is?
50. The complexity of searching an element from a set of n elements using Binary search algorithm is?

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

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

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

void inorder(struct node* root){
    if(root == NULL) return;
    inorder(root->left);
    printf("%d ->", root->data);
    inorder(root->right);
}

void preorder(struct node* root){
    if(root == NULL) return;
    printf("%d ->", root->data);
    preorder(root->left);
    preorder(root->right);
}

void postorder(struct node* root) {
    if(root == NULL) return;
    postorder(root->left);
    postorder(root->right);
    printf("%d ->", root->data);
}


struct node* createNode(value){
    struct node* newNode = malloc(sizeof(struct node));
    newNode->data = value;
    newNode->left = NULL;
```

```
    newNode->right = NULL;

    return newNode;
}

struct node* insertLeft(struct node *root, int value) {
    root->left = createNode(value);
    return root->left;
}


struct node* insertRight(struct node *root, int value){
    root->right = createNode(value);
    return root->right;
}


int main(){
    struct node* root = createNode(1);
    insertLeft(root, 12);
    insertRight(root, 9);

    insertLeft(root->left, 5);
    insertRight(root->left, 6);

    printf("Inorder traversal \n");
    inorder(root);

    printf("\nPreorder traversal \n");
    preorder(root);

    printf("\nPostorder traversal \n");
    postorder(root);
}
```

Sample Output:

```
Inorder traversal
```

```
5 ->12 ->6 ->1 ->9 ->

Preorder traversal

1 ->12 ->5 ->6 ->9 ->

Postorder traversal

5 ->6 ->12 ->9 ->1 ->
```

viva-questions

1.What is the space complexity of the in-order traversal in the recursive fashion? (d is the tree depth and n is the number of nodes

2.In postorder traversal of binary tree right subtree is traversed before visiting_____.

3.What is the possible number of binary trees that can be created with 3 nodes, giving the sequence N, M, L when traversed in post-order?

4.The post-order traversal of a binary tree is O P Q R S T. Then possible pre-order traversal will be?

5. A binary search tree contains values 7, 8, 13, 26, 35, 40, 70, 75. Which one of the following is a valid post-order sequence of the tree provided the pre-order sequence as 35, 13, 7, 8, 26, 70, 40 and 75?

6. Which of the following pair's traversals on a binary tree can build the tree uniquely?

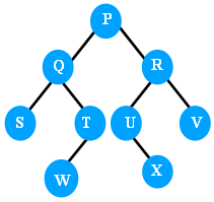
7.A full binary tree can be generated using__

8.The maximum number of nodes in a tree for which post-order and pre-order traversals may be equal is _____

9. The steps for finding post-order traversal are_____.

10. The pre-order and in-order are traversals of a binary tree are T M L N P O Q and L M N T O P Q. Which of following is post-order traversal of the tree?

11 For a binary tree the first node visited in in-order and post-order traversal is same. Find the postorder traversal of the binary tree shown below.



12 What is the space complexity of the post-order traversal in the recursive fashion? (d is the tree depth and n is the number of nodes)

13 To obtain a prefix expression, which of the tree traversals is used?

14 What is the space complexity of the in-order traversal in the recursive fashion? (d is the tree depth and n is the number of nodes)

15 What is the time complexity of level order traversal?

16 Which of the following graph traversals closely imitates level order traversal of a binary tree?

17 In a binary search tree, which of the following traversals would print the numbers in the ascending order?

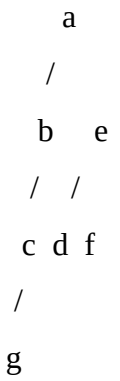
18. What is common in three different types of traversals (Inorder, Preorder and Postorder)?

19. The inorder and preorder traversal of a binary tree are d b e a f c g and a b d e c f g, respectively. The postorder traversal of the binary tree is:

20. Which traversal of tree resembles the breadth first search of the graph?

21. Which of the following tree traversal uses a queue data structure?

22. Which of the following cannot generate the full binary tree?
23. The value returned by the function DoSomething when a pointer to the root of a non-empty tree is passed as argument is (GATE CS 2004)
24. The array representation of a complete binary tree contains the data in sorted order. Which traversal of the tree will produce the data in sorted form?
25. Which of the following sequences denotes the post order traversal sequence of the given tree?



26. The post-order traversal of a binary search tree is given by 2, 7, 6, 10, 9, 8, 15, 17, 20, 19, 16, 12. Then the pre-order traversal of this tree is:
27. he in-order and pre-order traversal of a binary tree are d b e a f c g and a b d e c f g respectively. The post order traversal of a binary tree is
28. The number of rotations required to insert a sequence of elements 9,6,5,8,7,10 into an empty AVL tree is?
29. The inorder and preorder Traversal of binary Tree are dbeafcg and abdecfg respectively. The post-order Traversal is_____.
30. Level order Traversal of a rooted Tree can be done by starting from root and performing:
31. The in-order traversal of a tree resulted in FBGADCE. Then the pre-order traversal of that tree would result in
32. Suppose the numbers 7, 5, 1, 8, 3, 6, 0, 9, 4, 2 are inserted in that order into an initially empty binary search tree. The binary search tree uses the usual ordering on natural numbers. What is the inorder traversal sequence of the resultant tree?

33. What is tree traversal?
34 .What is binary tree?
35 .Whats is pre- order?
36 .What is post order?
37 .What is root?
38 .What is degree?
39 .How many binary trees are there?
40 .What is thread?
41 .What is strictly binary tree?
42 .What is skewed binary tree?
43 .What is complete binary tree?
44 .What is full binary tree?
45 .What is difference between complete full binary trees?
46 .Define forest.
47 .Write any three proprieties of binary tree?
48 .What is ADT?
49 .What is the use of bintree(bt)
50 . In how many ways a binary tree can be represented?

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

```
#include<stdio.h>

#include<stdlib.h>
void dfs();                         // For Deapth First Search(DFS) Traversal
void bfs();                         // For Breadth First Search(BFS) Traversal
struct link                         // Structure for adjacency list
{
        struct node *next;
        struct link *adj;
};

struct node                         // Structure for elements in the graph
{
        int data,status;
        struct node *next;
        struct link *adj;
};

struct node *start,*p,*q;           //Declaring pointer variable for structure node
struct link *l,*k;                  //Declaring pointer variable for structure node

void create()                       // Function to Create the graph
{
        int dat,flag=0;dat=1;               //Initializing 'flag' & 'dat' with a value of '0' &                  //'start' pointing to
NULL, as no value is being input till nowprintf("Enter the nodes in the graph(0 to end): ");
while(dat)
{
scanf("%d",&dat);                   //Getting the data from user
 if(dat==0)                         //If data entered is '0' then assume its the end of inputting elements
break;
p=(struct node*)malloc(sizeof(struct node)); //reserving memory space for nodal element p-
>data=dat;                          //storing the input data into node's data element
p->status=0;
p- >next=NULL;                      //next element is set to NULL p->adj=NULL;                  //previous
element is set to NULL
 if(flag==0)                        //If flag's value is zero then follow below procedure
{
start=p;
q=p;
flag++;
}                                   //If flag's value is not zero then follow below methodelse
{
q- >next=p;
q=p;
}
}                                   //Finishing the data entry
loopp=start;                        //Assigning the pointer 'p' the starting locationwhile(p!=NULL)
{
printf("Enter the links to %d (0 to end) : ",p->data);
flag=0;                             //Setting the initial value of 'flag' be '0'
while(1)
{
scanf("%d",&dat);
if(dat==0)
break;
k=(struct link*)malloc(sizeof(struct link));    //Allocating memory space for "link" element k->adj=NULL;
```

```
if(flag==0)
{
p->adj=k;
l=k;
flag++;
}
else
{
l->adj=k;
l=k;
}
q=start;
while(q!=NULL)
{
if(q->data==dat)
k->next=q;
q=q->next;
}
}
p=p->next;
}
}

void bfs()                  //Function for Breadth First Traversal of Graph
{
int q[20],i=1,j=0;
p=start;
while(p!=NULL)
{
p->status=0;
p=p->next;
}
p=start;
q[0]=p->data;
p->status=1;
while(1)
{
if(q[j]==0)
break;
p=start;
while(p!=NULL)
{
if(p->data==q[j])
break;
p=p->next;
}
k=p->adj;
while(k!=NULL)
{
q=k->next;
if(q->status==0)
{
q[i]=q->data;
q->status=1;
q[i+1]=0;
i++;
}
k=k->adj;
}
j++;
}
j=0;
printf("Breadth First Search Results\n");
while(q[j]!=0)                          //For printing the BFS result array
{
printf("%d ",q[j]);
```

```
j++;
}
getche();

}//Function for Depth First Search(BFS) Traversal.
void dfs()
{
int stack[20],top=1;
printf("Deapth First Search Results");
p=start;
while(p!=NULL)
{
p->status=0;
p=p->next;
}
p=start;
stack[0]=0;
stack[top]=p->data;
p->status=1;
while(1)
{
if(stack[top]==0)
break;
p=start;
while(p!=NULL)
{
if(p->data==stack[top])
break;
p=p->next;
}
printf("%d ",stack[top]);       //Printing the DFS result
 top--;
k=p->adj;
while(k!=NULL)
{
q=k->next;
if(q->status==0)
{
top++;
stack[top]=q->data;
q->status=1;
}
k=k->adj;
}
}
}

int main()
{
int ch;
create();                       //Invoking the create function
while(1)
{
printf("1: DFS\n2: BSF\n0: Exit\nEnter your choice: ");
scanf("%d",&ch);          //User enters his choice
switch(ch)
{
case 1:
dfs();                          //For Depth First Traversal
break;
case 2:
bfs();                          //For Breadth First Traversal
break;
case 0:
exit(0);                        //If Zero then exit the program (Omit this line if you don't want that)
```

```
break;
default:
printf("Incorrect choice!");
}
}
return 0;
}                    //End of program.
```

viva-questions

1What is Graph?

2 .What is undirected Graph?

3 .Define directed graph?

4 .What is BFS?

5 .What is spaning tree?

6 .The cost of the spaning tree is? 7 .What is krukskals algorithm? 8 .Explain prims algorithm?

9 .Breadth First Search is equivalent to which of the traversal in the Binary Trees?

10. The Breadth First Search traversal of a graph will result into?

11 . In BFS, how many times a node is visited?

12 .What can be the applications of Breadth First Search?

13. When the Breadth First Search of a graph is unique?

14 .Depth First Search is equivalent to which of the traversal in the Binary Trees?

15 .The Depth First Search traversal of a graph will result into?

16 .What can be the applications of Depth First Search?

17 .When the Depth First Search of a graph is unique?

18 .In Depth First Search, how many times a node is visited?

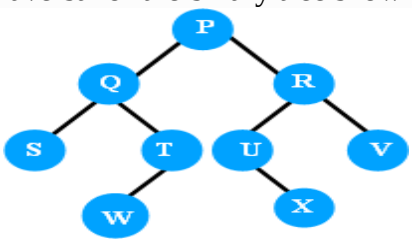
19 .What is DFS?

20 .The aim of DFS algorithm is?

21 .The aim of BFS algorithm is?

22 .Traversal of a graph is different from tree because?

23 .What is BFS explain with example?
24 .What is DFS explain with example?
25 .Is BFS is faster then DFS?
26 .What is the time complexity of BFS and DFS?
27 .What is BFS and DFS in c?
28 .Does DFS find short path?
29 .What is BFS and DFS tree?
30 .Why DFS is not optimal?
31 .Traversal of a graph is different from tree because?
32 .What is common in three different types of traversals (Inorder, Preorder and Postorder)?
33 .Which traversal of tree resembles the breadth first search of the graph?
34 .Level order Traversal of a rooted Tree can be done by starting from root and performing?
35 .The Data structure used in standard implementation of Breadth First Search is?
36 .What is the time complexity of level order traversal?
37 .What is the space complexity of the in-order traversal in the recursive fashion?
38 .What is the time complexity of pre-order traversal in the iterative fashion?
39 .What is the space complexity of the post-order traversal in the recursive fashion?
40 .To obtain a prefix expression, which of the tree traversals is used?
41 .A full binary tree can be generated using?
42 .The maximum number of nodes in a tree for which post-order and pre-order traversals may be equal is?
43 .For a binary tree the first node visited in in-order and post-order traversal is same?
44 .Find the postorder traversal of the binary tree shown below.



45 .What is pre order?

46 .What is post order?

47 .What is in order?

48 .What is graph Traversal ?

49 .Difference between pre order, post order, in order?

50 .Which algorithms can be used to most efficiently determine the presence of a cycle?