GLOBAL ACADEMY OF TECHNOLOGY

Autonomous Institute affiliated to VTU, Belagavi,
Accredited by NAAC with 'A' grade
Ideal Homes Township, RR Nagar, Bengaluru – 560098

Department of Computer Science and Engineering (AI&ML)

Laboratory Manual

Data Structure Laboratory
Course Code: BCIL24305

III Semester
Academic Year 2025-26

Prepared By
Dr. Shruti B V, Associate Professor
Prof. Sharadadevi K, Assistant Professor

Approved By
Professor & Head,
Dept. of CSE(AI&ML)

LABORATORY CERTIFICATE

This is to certify that Mr./Ms. _____ bearing USN_____ of the Department CSE (AIML) has satisfactorily completed the course on **Data Structures Laboratory – BCIL24305** prescribed by Global Academy of Technology (Autonomous Institute, under VTU), Bengaluru for the Academic Year 2025-26.

MARKS	
MAXIMUM	**OBTAINED**
50	

Date:

Signature of the Faculty **Signature of the HOD**

DOCUMENT LOG

Name of the document	Data Structure Laboratory Manual
Scheme	2025
Current version number and date	Version 1.0 / 01.08.2025
Subject code	BCIL24305
Prepared by	Dr. Shruti B V. Prof. Sharadadevi K.
Approved by	HOD, Dept. of CSE(AI&ML)

Table of Contents

Vision of the Institute

Become a premier institution imparting quality education in engineering and management tomeet the changing needs of society.

Mission of the Institute

M1: Create environment conductive for continuous learning through quality teaching and learning processes supported by modern infrastructure.

M2: Promote research and innovation through collaboration with industries.

M3: Inculcate ethical values and environmental conscious through holistic education programs.

Vision of the Department

To Become a leading hub of excellence in education, research in the field of Computer Science and Engineering providing AI-driven solutions with holistic development for the needs of the Society.

Mission of the Department

- To Provide aspiring engineers for industry and academia by offering excellent education in emerging AI techniques.
- To equip value-added technical and research-oriented education by satisfying societal needs
- To educate with professional integrity values and ethics for environment awareness.

PROGRAM EDUCATIONAL OBJECTIVES(PEOs)

- **PEO1:** Design and develop innovative intelligent systems for the welfare of Society.
- **PEO2: Engage in lifelong learning process through higher education and to inculcate innovative ideas in research field.**
- **PEO3:** Lead the IT Industry with management and Entrepreneurship skills.

PROGRAM SPECIFIC OUTCOMES(PSOs)

- **PSO1: To Produce graduates with industry-ready skills with the knowledge of Computer Science & Machine Learning technology based to solve real world problems.**
- **PSO2:** Ability to develop many successful applications and designing efficient algorithms for intelligent systems within the realm of artificial intelligence incorporating in data analytics, Natural language processing and Internet of Things.

PROGRAM OUTCOMES (PO's)

Engineering Graduates will be able to:

PO1: Engineering Knowledge:
Apply knowledge of mathematics, natural science, computing, engineering fundamentals and an engineering specialization as specified in WK1 to WK4 respectively to develop to the solution of complex engineering problems.

PO2: Problem Analysis:
Identify, formulate, review research literature and analyze complex engineering problems reaching substantiated conclusions with consideration for sustainable
development (WK1 to WK4).

PO3: Design/Development of Solutions:
Design creative solutions for complex engineering problems and design/develop systems/components/processes to meet identified needs with consideration for the public health and safety, whole-life cost, net zero carbon, culture, society and environment as required (WK5).

PO4: Conduct Investigations of Complex Problems:
Conduct investigations of complex engineering problems using research-based knowledge including design of experiments, modelling, analysis & interpretation of data to provide valid conclusions (WK8).

PO5: Engineering Tool Usage:
Create, select, and apply appropriate techniques, resources, and modern engineering & IT tools, including prediction, and modelling recognizing their limitations to solve complex engineering problems (WK2 and WK6).

PO6: The Engineer and The World:
Analyse and evaluate societal and environmental aspects while solving complex engineering problems for its impact on sustainability with reference to economy, health, safety, legal framework, culture and environment (WK1, WK5, and WK7).

PO7: Ethics:
Apply ethical principles and commit to professional ethics, human values, diversity, and inclusion; adhere to national & international laws (WK9).

PO8: Individual and Collaborative Teamwork:
Function effectively as an individual, and as a member or leader in diverse/multi-disciplinary teams.

PO9: Communication:
Communicate effectively and inclusively within the engineering community and society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations considering cultural, language, and learning differences.

PO10: Project Management and Finance:
Apply knowledge and understanding of engineering management principles and economic decision-making and apply these to one's own work, as a member and leader in a team, and to manage projects and in multidisciplinary environments.

PO11: Life-Long Learning:
Recognize the need for, and have the preparation and ability for i) independent and life-long learning ii) adaptability to new and emerging technologies, and iii) critical thinking in the broadest context of technological change (WK8).

COURSE OUTCOMES:

Upon successful completion of this course, students are able to:

CO1	Implement stack and queue operations using array.
CO2	Demonstrate Recursive functions.
CO3	Demonstrate the working of Linked Lists.
CO4	Implement Binary tree traversals.
CO5	Implement priority queue, heap and hashing.

CO-PO Mapping

POs → / COs ↓	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PSO1	PSO2
CO1	3	3	3	-	2	-	-	2	-	2	-	2	-
CO2	3	3	3	-	2	-	-	2	-	2	-	2	-
CO3	3	3	3	-	2	-	-	2	-	2	-	2	-
CO4	3	3	3	-	2	-	-	2	-	2	-	2	-
CO5	3	3	3	-	2	-	-	2	-	2	-	2	-

Lab Evaluation Sheet

SNO	PARAMETERS	MARKS
1.	RECORD	10
2.	MINI PROJECT	20
3.	LAB TEST	20

Lab Record Evaluation Rubrics

Attribute	Max Marks	Good (5-6)	Satisfactory (3-4)	Poor (0-2)
Write-Up + Execution	06	• Debugs the program independently. • Executed the program for all possible inputs. • Record of output properly in observation book	• Works with little help from faculty. • All possible input cases are not covered. • Incomplete record of output in observation book	Unable to complete the execution of the program within the lab session.
Attribute	**Max Marks**	**Good (4)**	**Satisfactory (2-3)**	**Poor (0-1)**
Viva Voce	04	• Able to explain the logic of the program. • Answered all questions.	• logic of the program. • Answered all questions. • Partially understood the logic of the program. • Answered few questions.	• Not understood the logic of the program. • Not answering any questions.

Rubrics for Evaluation of Internal Assessment Test

Attribute	Max Marks	Good (4)	Satisfactory (2-3)	Poor (0-1)
Write-Up	04	• Complete program without errors written. • Input and expected output for all test cases.	• Complete program without errors written. • Input and expected output not for all test cases.	• Incomplete code. • Indentation is missing. • All test cases not covered.
Attribute	**Max Marks**	**Good (9-12)**	**Satisfactory (5-8)**	**Poor (0-4)**
Execution	12	• Debugs the program independently. • Executed the program for all possible inputs. Record of output properly in observation	• Works with little help from faculty. • All possible input cases are not covered. Incomplete record of output in observation book	• Unable to complete the execution of the program within the lab session.

		book		
Attribute	Max Marks	Good (4)	Satisfactory (2-3)	Poor (0-1)
Viva Voce	04	• Able to explain the logic of the program. • Answered all questions.	• logic of the program. • Answered all questions. • Partially understood the logic of the program. • Answered few questions.	• Not understood the logic of the program. Not answering any questions.

Rubrics for Evaluation of Mini Project

Attribute	Max Marks	Good (4)	Satisfactory (2-3)	Poor (0-1)
Project Demonstration	08	• All defined objectives are achieved Each module working well and properly demonstrated • All modules of project are well integrated, and system working is accurate	• All defined objectives are achieved Each module working well and properly demonstrated • Integration of all modules not done and system working is not veey satisfactory	• All defined objectives are achieved. Modules are working well in isolation and properly demonstrated • Modules of project are not properly integrated
Attribute	Max Marks	Good (7-8)	Satisfactory (4-6)	Poor (0-3)
Design Methodology	08	• Division of problem into modules and good selection of computing framework • Appropriate design methodology and properly justification	• Division of problem into modules and good selection of computing framework • Design methodology not properly justified	• Division of problem into modules but inappropriate selection of computing framework • Design methodology not defined properly
Attribute	Max Marks	Good (4)	Satisfactory (2-3)	Poor (0-1)
Report	04	• Project report is according to the specified format • References and citations are appropriate and well mentioned	• Project report is according to the specified format • References and citations are appropriate but not mentioned well	• Project report is according to the specified format but some mistakes • In-sufficient references and citations

Lab Evaluation Sheet

Sno	Date	Program	Writeup + Exe (6)	Viva Voce (4)	Total(10)
1.		Develop a menu driven Program for the following operations on STACK of Integers (Array Implementation of Stack with maximum size MAX) a. Push an Element on to Stack b. Pop an Element from Stack c. Demonstrate Overflow and Underflow situations on Stack d. Display the status of Stack e. Exit Support the program with appropriate functions for each of the above operations			
2.		Develop a Program for converting an Infix Expression to Postfix Expression. Program should support for both parenthesized and free parenthesized expressions with the operators: +, -, *, /, % (Remainder), ^ (Power) and alphanumeric operands			
3.		Develop and Implement a Program for evaluation of Stack Suffix expression with single digit operands and operators: +, -, *, /, %, ^.			
4.		Develop recursive program to i) To Find GCD of 2 numbers ii) To Solve the Tower of Hanoi Problem			
5.		Develop a menu driven Program for the following operations on QUEUE of Characters (Array Implementation of QUEUE with maximum size MAX) a. Enqueue an Element on to Queue b. Dequeue an Element from Queue c. Demonstrate Overflow and Underflow situations on Queue d. Display the status of Queue e. Exit Support the program with appropriate functions for each of the above operation.			
6.		Implement a program to multiply two polynomials using singly linked list.			
7.		Design a doubly linked list to represent sparse matrix. Each node in the list can have the row and column index of the matrix element and the value of the element. Print the complete matrix as the output.			
8.		Write a program to create Binary Tree and to traverse the tree using In-order, Preorder and Post order.			
9.		Write a program to implement priority queue using Heap.			
10.		Write a program to implement Hashing using Linear probing. Implement insertion, deletion, search and display.			
Average Marks of Lab Record			**/ 10**		

	MAX MARKS	OBTAINED MARKS	SIGNATURE OF FACULTY IN-CHARGE
RECORD	10		1. Dr. Shruti B V.
MINI PROJECT	20		
LAB TEST	20		2. Prof. Sharadadevi K.
FINAL	50		

CHAPTER – 1

INTRODUCTION

Data Structure is defined as the way in which data is organized in the memory location. There are two types of data structures as shown in Fig1.1.

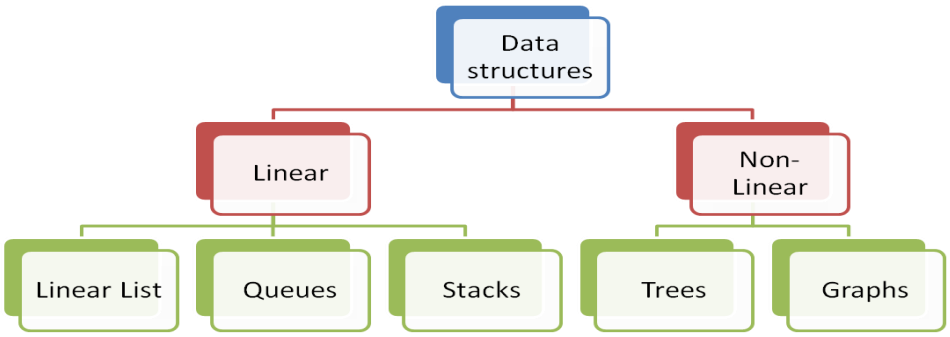

Fig 1.1: Types of Data Structures

Linear Data Structure:

In linear data structure all the data are stored linearly or contiguously in the memory. All the data are saved in continuously memory locations and hence all data elements are saved in one boundary. A linear data structure is one in which we can reach directly only one element from another while travelling sequentially. The main advantage is that we find the first element, and then it's easy to find the next data elements. The disadvantage, the size of the array must be known before allocating the memory.

The different types of linear data structures are:

- Arrays
- Stacks
- Queues
- Linked Lists

Non-Linear Data Structure: Every data item is attached to several other data items in a way that is specific for reflecting relationships. The data items are not arranged in a sequential structure. The various types of nonlinear data structures are:

- Trees
- Graphs

CHAPTER - 2

SYLLABUS PROGRAMS

1.Develop a menu driven Program in C for the following operations on STACK of Integers (Array Implementation of Stack with maximum size MAX)

a. Push an Element on to Stack

b. Pop an Element from Stack

c. Demonstrate Overflow and Underflow situations on Stack

d. Display the status of Stack

e. Exit

Support the program with appropriate functions for each of the above operations

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 5
struct stack
{
   int top;
   int data[SIZE];
};
typedef struct stack STACK;

void push(STACK *s,int item)
{
   if(s->top==SIZE-1)
      printf("\n Stack Overflow");
   else
   {
      s->top=s->top+1;
      s->data[s->top]=item;
   }
```

```
}

void pop(STACK *s)
{
  if(s->top==-1)
      printf("\n Stack Underflow");
  else
   {
      printf("\n Element popped is %d",s->data[s->top]);
      s->top=s->top-1;
   }
}

void display(STACK s)
{
   int i;
   if(s.top ==-1)
      printf("\n Stack Empty");
   else
   {
      printf("\ Stack content are\n");
      for(i=s.top;i>=0;i--)
          printf("%d\n",s.data[i]);
   }
}
int main()
{
   int ch,item;
   STACK s;
   s.top=-1;
```

```
    for(;;)
    {
    printf("STACK OPERATIONS:");
    printf("\n1. PUSH\n2. POP\n3. DISPLAY\n4.EXIT\n");
    printf("\nRead Choice :");
    scanf("%d",&ch);
    switch(ch)
    {
       case 1:printf("\n Read element to be pushed :");
              scanf("%d",&item);
              push(&s,item);
              break;
       case 2:pop(&s);
               break;
       case 3:display(s);
               break;
       default:exit(0);
    }
    }
    return 0;

}
```

OUTPUT:STACK OPERATIONS:

1.PUSH 2.POP 3.DISPLAY 4.EXIT

Enter your choice:1

Enter element to be push:5

Pushed 5 on to the stack

Enter your choice:1

Enter the elements to push:10

Pushed 10 on to the stack

Enter your choice:1

Enter the element to be pushed:15

Pushed 15 onto the stack

Enter the choice :1

Enter the element to be push:20

Pushed the element 20 on to the stack

Enter the choice:3

Stack contains:5 10 15 20

Enter the choice:2

Popped 20 from the stack

Enter the choice:3

Stack contains:5 10 15

2.Develop a Program in C for converting an Infix Expression to Postfix Expression. Program should support for both parenthesized and free parenthesized expressions with the operators: +, -, *, /, % (Remainder), ^ (Power) and alphanumeric operands.

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#define SIZE 20
struct stack
{
   int top;
   char data[SIZE];
};
typedef struct stack STACK;
void push(STACK *s,char item)
{
   s->data[++(s->top)]=item;
}

char pop(STACK *s)
{
   return s->data[(s->top)--];
}

int preced(char symbol)
{
   switch(symbol)
   {
      case '^':return 5;
      case '*':
        case '/':return 3;
```

```
        case '+':
          case '-':return 1;
    }
}

void infixtopostfix(STACK *s,char infix[SIZE])
{
    int i,j=0;
    char postfix[SIZE],temp,symbol;
    for(i=0;infix[i]!='\0';i++)
    {
        symbol=infix[i];
        if(isalnum(symbol))
            postfix[j++]=symbol;
        else
        {
            switch(symbol)
            {
                case '(':push(s,symbol);
                        break;
                case ')':temp=pop(s);
                        while(temp!='(')
                        {
                            postfix[j++]=temp;
                            temp=pop(s);
                        }
                        break;
                case '+':
                case '-':
                case '*':
```

```
            case '/':
            case '^': if (s->top ==-1 || s->data[s->top]=='(')
                        push(s,symbol);
                    else
                    {
                       while(preced(s->data[s->top])>= preced(symbol) && s->top!=-1 &&s->data[s->top]!='(')
                            postfix[j++]=pop(s);
                       push(s,symbol);
                    }
                    break;
            default :printf("\n Invalid!!!!!");
                     exit(0);

          }
       }
    }
    while(s->top!=-1)
       postfix[j++]=pop(s);
    postfix[j]='\0';
    printf("\n The postfix expression is %s\n",postfix);
}
int main()
{
    STACK s;
    s.top=-1;
    char infix[SIZE];
    printf("\n Read Infix expression\n");
    scanf("%s",infix);
    infixtopostfix(&s,infix);
    return 0;
```

```
}
```

OUTPUT:

Enter an infix expression:A-(B/C+(D%E*F)/G*H

The postfix expression is:ABC/DE%F*G/H*

3.Develop and Implement a Program for evaluation of Stack Suffix expression with single digit operands and operators: +, -, *, /, %, ^.

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <math.h>
#define SIZE 20
struct stack
{
   int top;
   float data[SIZE];
};
typedef struct stack STACK;
void push(STACK *s,float item)
{
   s->data[++(s->top)]=item;
}

float pop(STACK *s)
{
   return s->data[(s->top)--];
}

float operate(float op1,float op2,char symbol)
{
   switch(symbol)
   {
      case '+':return op1+op2;
      case '-':return op1-op2;
      case '*':return op1*op2;
```

```
        case '/':return op1/op2;
        case '^':return pow(op1,op2);
    }
}

float eval(STACK *s,char postfix[SIZE])
{
    int i;
    char symbol;
    float res,op1,op2;
    for(i=0;postfix[i]!='\0';i++)
    {
        symbol=postfix[i];
        if(isdigit(symbol))
            push(s,symbol-'0');
        else
        {
            op2=pop(s);
            op1=pop(s);
            res=operate(op1,op2,symbol);
            push(s,res);
        }
    }
    return pop(s);
}

int main()
{
    char postfix[SIZE];
    STACK s;
```

```
    float ans;
    s.top=-1;
    printf("\n Read postfix expr\n");
    scanf("%s",postfix);
    ans=eval(&s,postfix);
    printf("\n The final answer is %f\n",ans);
    return 0;
}
```

OUTPUT:

Enter the Postfix Expression(single digit operands):934*8+4/-

The final answer is 4.0000

4.Develop recursive program in C to

i) To Find GCD of 2 numbers

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

int gcd(int a,int b)
{
   if(b!=0)
      return gcd(b,a % b);
   return a;
}
int main()
{
   int a,b;
   printf("\n Read two numbers:");
   scanf("%d%d",&a,&b);
   printf("\n GCD of %d and %d is %d\n",a,b,gcd(a,b));
   return 0;
}
```

OUTPUT:

Enter two numbers to find GCD:10 4

GCD of 10 and 4 is 2.

ii) To Solve the Tower of Hanoi Problem.

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

void towerofHanoi(int n,char source,char temp,char destination)
```

```
{
   if(n==1)
      printf("\n Move %d disc from %c to %c",n,source,destination);
   else
   {
      towerofHanoi(n-1,source,destination,temp);
      printf("\n Move %d disc from %c to %c",n,source,destination);
      towerofHanoi(n-1,temp,source,destination);
   }
}

int main()
{
   int n;
   printf("\n Read number of discs:");
   scanf("%d",&n);
   towerofHanoi(n,'S','T','D');
   return 0;
}
```

OUTPUT:

```
Enter no.of discs:3
Move 1 disc from A to C
Move 2 disc from A to B
Move 1 disc from C to B
Move 3 disc from A to C
Move 1 disc from B to A
Move 2 disc from B to C
Move 1 disc from A to C
```

5. Develop a menu driven Program in C for the following operations on QUEUE of Characters (Array Implementation of QUEUE with maximum size MAX)

a. Enqueue an Element on to Queue

b. Dequeue an Element from Queue

c. Demonstrate Overflow and Underflow situations on Queue

d. Display the status of Queue

e. Exit

Support the program with appropriate functions for each of the above operations

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 5
struct queue
{
   int front,rear;
   char data[SIZE];
};
typedef struct queue QUEUE;

void enqueue(QUEUE *q,char item)
{
   if(q->rear==SIZE-1)
      printf("\n Queue full");
   else
   {
      q->rear=q->rear+1;
      q->data[q->rear]=item;
      if(q->front==-1)
         q->front=0;
   }
}
```

```
char dequeue(QUEUE *q)
{
   int del;
   if(q->front==-1)
   {
      printf("\n Queue empty");
      return -1;
   }
   else
   {
      del=q->data[q->front];
      if(q->front==q->rear)
      {
         q->front=-1;
         q->rear=-1;
      }
      else
         q->front=q->front+1;
      return del;
   }
}

void display(QUEUE q)
{
   int i;
   if(q.front==-1)
      printf("\n Queue Empty");
   else
   {
```

```
        printf("\n Queue content are\n");
        for(i=q.front;i<=q.rear;i++)
            printf("%c\t",q.data[i]);
    }
}
int main()
{
    char item,del;
    int ch;
    QUEUE q;
    q.front=-1;
    q.rear=-1;
    for(;;)
    {
    printf("QUEUE OPERATIONS:");
    printf("\n1. Enqueue\n2. Dequeue\n3. Display\n4.Exit");
    printf("\nRead Choice :");
    scanf("%d",&ch);
    getchar();
    switch(ch)
    {
        case 1:printf("\n Read element to be inserted :");
               scanf("%c",&item);
               enqueue(&q,item);
               break;
        case 2:del=dequeue(&q);
               if(del!=-1)
                   printf("\n Element deleted is %c\n",del);
               break;
        case 3:display(q);
```

```
            break;
        default:exit(0);
    }
    }
    return 0;
}
```

OUTPUT:QUEUE OPERATIONS:

1.Enqueue 2.Dequeue 3.Display

Enter your choice:1

Enter the element to enqueue:5

Enqueued 5 to queue

Enter your choice:1

Enter the element to enqueue:10

Enqueued 10 to queue.

Enter your choice:1

Enter the element to enqueue:15

Enqueued 15 to queue.

Enter your choice:1

Enter the element to enqueue:20

Enqueued 20 to queue.

Enter your choice:1

Enter the element to enqueue:25

Enqueued 25 to queue.

Enter your choice:1

Enter the element to enqueue:30

Stack overflow! Cannot enqueue.

Enter your choice:2

Dequeued 5 from queue

Enter your choice:3

Queue contents are:10 15 20 25

6.Implement a program to multiply two polynomials using singly linked list.

```
#include <stdio.h>
#include <stdlib.h>
#define SIZE 5
int count;
struct node
{
   int co,po;
   struct node *addr;
};
typedef struct node *NODE;
NODE insertend(NODE start,int co,int po)
{
   NODE temp,cur;
   temp=(NODE)malloc(sizeof(struct node));
   temp->co=co;
   temp->po=po;
   temp->addr=NULL;
   if(start==NULL)
      return temp;
   cur=start;
   while(cur->addr!=NULL)
      cur=cur->addr;
   cur->addr=temp;
   return start;
}

void display(NODE start)
{
```

```
    NODE temp;
    if(start==NULL)
        printf("\n Polynomial Empty");
    else
    {
        temp=start;
        while(temp->addr!=NULL)
        {
            printf("%dx^%d+",temp->co,temp->po);
            temp=temp->addr;
        }
            printf("%dx^%d\n",temp->co,temp->po);
    }
}

NODE addterm(NODE res,int co,int po)
{
    NODE temp,cur;
    temp=(NODE)malloc(sizeof(struct node));
    temp->co=co;
    temp->po=po;
    temp->addr=NULL;
    if(res==NULL)
        return temp;
    cur=res;
    while(cur!=NULL)
    {
        if(cur->po==po)
        {
            cur->co=cur->co+co;
```

```
            return res;
        }
        cur=cur->addr;
    }
    if(cur==NULL)
        res=insertend(res,co,po);
    return res;
}
NODE multiply(NODE poly1,NODE poly2)
{
    NODE p1,p2,res=NULL;
    for(p1=poly1;p1!=NULL;p1=p1->addr)
        for(p2=poly2;p2!=NULL;p2=p2->addr)
            res=addterm(res,p1->co*p2->co,p1->po+p2->po);
    return res;
}
int main()
{
    NODE poly1=NULL,poly2=NULL,poly;
    int co,po;
    int i,n,m;
    printf("\nRead no of terms of first polynomial:");
    scanf("%d",&n);
    for(i=1;i<=n;i++)
    {
        printf("\n Read CO and PO of %d term : ",i);
        scanf("%d%d",&co,&po);
        poly1=insertend(poly1,co,po);
    }
    printf("\n First polynomial is\n");
```

```
    display(poly1);
    printf("\nRead no of terms of second polynomial:");
    scanf("%d",&m);
    for(i=1;i<=m;i++)
    {
       printf("\n Read CO and PO of %d term : ",i);
       scanf("%d%d",&co,&po);
       poly2=insertend(poly2,co,po);
    }
    printf("\n Second polynomial is\n");
    display(poly2);
    poly=multiply(poly1,poly2);
    printf("\n Resultant polynomial is\n");
    display(poly);
    return 0;

}
```

OUTPUT:

```
Read the no.of terms in polynomil1:3
Read the co-efficient and power of 1 term:4 2
Read the co-efficient and power of 2 term:2 1
Read the co-efficient and power of 1 term:1 0
Read the no.of terms in polynomil1:2
Read the co-efficient and power of 2 term:3 1
Read the co-efficient and power of 2 term:1 0

First polynomial is:4*x^2+2*x^1+1*x^0
Second polynomial is:3*x^1+1*x^0

Resultant polynomial is:12*x^3+10*x^2+5*x^2+5*x^1+1*x^0
```

7.Design a doubly linked list to represent sparse matrix. Each node in the list can have the row and column index of the matrix element and the value

of the element. Print the complete matrix as the output.

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

struct node
{
   int row,col,data;
   struct node *next;
   struct node *prev;
};
typedef struct node *NODE;

NODE insertend(NODE start,int row,int col,int item)
{
   NODE temp,cur;
   temp=(NODE)malloc(sizeof(struct node));
   temp->row=row;
   temp->col=col;
   temp->data=item;
   temp->next=NULL;
   temp->prev=NULL;
   if(start == NULL)
      return temp;
   cur=start;
   while(cur->next!=NULL)
      cur = cur->next;
   cur->next=temp;
   temp->prev=cur;
```

```
    return start;
}

void display(NODE start)
{
    NODE temp;
    if(start==NULL)
        printf("\n list is empty");
    else
    {
        printf("\nROW\tCOL\tDATA\n");
        temp=start;
        while(temp!=NULL)
        {
            printf("%d\t%d\t%d\n",temp->row,temp->col,temp->data);
            temp=temp->next;
        }
    }
}

void displaymatrix(NODE start,int m,int n)
{
    NODE temp;
    int i,j;
    temp=start;
    printf("\n The Sparse matrix is\n");
    for(i=1;i<=m;i++)
        {
            for(j=1;j<=n;j++)
            {
```

```
            if(temp!=NULL && temp->row == i && temp->col == j)
            {
               printf("%d\t",temp->data);
               temp=temp->next;
            }
            else
               printf("0\t");
      }
     printf("\n");
    }

}



int main()
{
   NODE start = NULL;
   int i,j,m,n,item;
   printf("\n Read the order of the matrix\n");
   scanf("%d%d",&m,&n);
   printf("\n Read the matrix\n");
   for(i=1;i<=m;i++)
   {
      for(j=1;j<=n;j++)
      {
         scanf("%d",&item);
         if(item!=0)
            start=insertend(start,i,j,item);
       }
   }
```

```
    display(start);
    displaymatrix(start,m,n);
    return 0;
}
```

OUTPUT:

```
Read the order of the matrix:3 4
Read the elements of Matrix:
0
0
2
0
0
6
5
0
3
0
0
0
ROW COL DATA
1  3  2
2  3  6
2  4  5
3  1  1

THE SPARSE MATRIX IS:
0 0 2 0
0 6 5 0
3 0 0 0
```

8.Write a C program to create Binary Tree and to traverse the tree using In-order, Preorder and Post order.

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
   int data;
   struct node *left;
   struct node *right;
};
typedef struct node *NODE;

NODE create_node(int item)
{
   NODE temp;
   temp=(NODE)malloc(sizeof(struct node));
   temp->data=item;
   temp->left=NULL;
   temp->right=NULL;
   return temp;
}

NODE Insertbst(NODE root,int item)
{
   NODE temp;
   temp=create_node(item);
   if(root==NULL)
      return temp;
   else
```

```
    {
      if(item < root->data)
          root->left=Insertbst(root->left,item);
       else
          root->right=Insertbst(root->right,item);
    }
    return root;

}

void preorder(NODE root)
{
    if(root!=NULL)
    {
        printf("%d\t",root->data);
        preorder(root->left);
        preorder(root->right);
    }
}

void inorder(NODE root)
{
    if(root!=NULL)
    {
        inorder(root->left);
        printf("%d\t",root->data);
        inorder(root->right);
    }
}
```

```
void postorder(NODE root)
{
   if(root!=NULL)
   {
      postorder(root->left);
      postorder(root->right);
      printf("%d\t",root->data);
   }
}


int main()
{
   NODE root = NULL;
   int ch,item;
   for(;;)
   {
      printf("\n 1. Insert");
      printf("\n 2. Preorder");
      printf("\n 3. Inorder");
      printf("\n 4. Postorder");
      printf("\n 5. Exit");
      printf("\n Read ur choice:");
      scanf("%d",&ch);
      switch(ch)
      {
         case 1:printf("\n Read element to be inserted :");
            scanf("%d",&item);
            root=Insertbst(root,item);
            break;
```

```
            case 2:printf("\n The Preorder traversal is\n");
                preorder(root);
                break;
            case 3:printf("\n The Inorder traversal is\n");
                inorder(root);
                break;
            case 4:printf("\n The Postorder traversal is\n");
                postorder(root);
                break;
            default :exit(0);
        }
    }
    return 0;
}
```

OUTPUT:

```
1.Insert 2.Preorder 3.Inorder 4.Postorder 5.Exit
Read your choice:1
Read the element to be inserted:18
1.Insert 2.Preorder 3.Inorder 4.Postorder 5.Exit
Read your choice:1
Read the element to be inserted:4
1.Insert 2.Preorder 3.Inorder 4.Postorder 5.Exit
Read your choice:1
Read the element to be inserted:1
1.Insert 2.Preorder 3.Inorder 4.Postorder 5.Exit

Read your choice:1
Read the element to be inserted:0
1.Insert 2.Preorder 3.Inorder 4.Postorder 5.Exit
```

Read your choice:1

Read the element to be inserted:7

1.Insert 2.Preorder 3.Inorder 4.Postorder 5.Exit

Read your choice:1

Read the element to be inserted:12

1.Insert 2.Preorder 3.Inorder 4.Postorder 5.Exit

Read your choice:1

Read the element to be inserted:47

1.Insert 2.Preorder 3.Inorder 4.Postorder 5.Exit

Read your choice:1

Read the element to be inserted:21

1.Insert 2.Preorder 3.Inorder 4.Postorder 5.Exit

Read your choice:1

Read the element to be inserted:25

1.Insert 2.Preorder 3.Inorder 4.Postorder 5.Exit

Read your choice:1

Read the element to be inserted:90

1.Insert 2.Preorder 3.Inorder 4.Postorder 5.Exit

Read your choice:2

The Pre-order traversal is:

18,4,1,0,7,12,47,25,21,90

1.Insert 2.Preorder 3.Inorder 4.Postorder 5.Exit

Read your choice:3

The Inorder traversal is:

0,1,4,7,12,18,21,25,47,90

Read your choice:4

The Post order traversal is:

0,12,7,1,25,21,90,47,4,18

1.Insert 2.Preorder 3.Inorder 4.Postorder 5.Exit

Read your choice:5

9.Write a C program to implement priority queue using Heap.

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

void heapify(int a[10],int n)
{
   int i,k,v,j,flag=0;
   for(i=n/2;i>=1;i--)
   {
      k=i;
      v=a[k];
      while(!flag && 2*k <= n)
      {
         j=2*k;
         if(j<n)
         {
            if(a[j]<a[j+1])
               j=j+1;
         }
         if(v>=a[j])
            flag=1;
         else
         {
            a[k]=a[j];
            k=j;
         }
      }
      a[k]=v;
      flag=0;
```

```
    }
}

int main()
{
    int n,i,a[10],ch;
    for(;;)
    {
        printf("\n 1. Create Heap");
        printf("\n 2. Extractmax");
        printf("\n 3. Exit");
        printf("\n Read Choice :");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:printf("\n Read no of elements :");
                   scanf("%d",&n);
                   printf("\n Read Elements\n");
                   for(i=1;i<=n;i++)
                    scanf("%d",&a[i]);
                   heapify(a,n);
                   printf("\n Elements after heap\n");
                   for(i=1;i<=n;i++)
                    printf("%d\t",a[i]);
                    break;
            case 2:if(n>=1)
                   {
                      printf("\n Element deleted is %d\n",a[1]);
                      a[1]=a[n];
                      n=n-1;
```

```
                heapify(a,n);
                if(n!=0)
                {
                   printf("\n Elements after reconstructing heap\n");
                   for(i=1;i<=n;i++)
                      printf("%d\t",a[i]);
                }
             }
             else
                printf("\n No element to delete");
             break;
       default:exit(0);

     }
   }
   return 0;
}
```

OUTPUT:

1.Create Heap 2.Extract Max 3.Exit

Read Choice:1

Read no.of elements:5

Elements after heap:

40 30 15 10 20

1.Create Heap 2.Extract Max 3.Exit

Read Choice:2

Element deleted is 40

Elements after reconstructing heap:

30 20 15 10

1.Create Heap 2.Extract Max 3.Exit

Read Choice:3

10.Write a C program to implement Hashing using Linear probing. Implement insertion, deletion, search and display.

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

int h[TABLE_SIZE]={NULL};

void insert()
{
 int key,index,i,flag=0,hkey;
 printf("\nenter a value to insert into hash table:");
 scanf("%d",&key);
 hkey=key%TABLE_SIZE;
 for(i=0;i<TABLE_SIZE;i++)
   {
    index=(hkey+i)%TABLE_SIZE;
    if(h[index] == NULL)
    {
      h[index]=key;
       break;
    }
   }
   if(i == TABLE_SIZE)
      printf("\nelement cannot be inserted\n");
}

void search()
{
```

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

void display()
{
 int i;
 printf("\nelements in the hash table are \n");
 for(i=0;i< TABLE_SIZE; i++)
     printf("\nat index %d \t value =  %d",i,h[i]);

}
main()
{
   int ch,i;
   for(;;)
   {
```

```
        printf("\n1.Insert\n2.Display\n3.Search\n4.Exit\n");
        printf("\n Read Choice :");
        scanf("%d",&ch);
        switch(ch)
        {
          case 1:
             insert();
             break;
          case 2:
             display();
             break;
          case 3:
             search();
             break;
          default:exit(0);
        }
      }
}
```

OUTPUT:

1.Insert 2.Display 3.Read 4.Exit

Read choice:1

Enter the value in the hash table:1

1.Insert 2.Display 3.Read 4.Exit

Read choice:1

Enter the value in the hash table:44

1.Insert 2.Display 3.Read 4.Exit

Read choice:1

Enter the value in the hash table:67

1.Insert 2.Display 3.Read 4.Exit

Read choice:1

Enter the value in the hash table:22

1.Insert 2.Display 3.Read 4.Exit

Read choice:1

Enter the value in the hash table:58

1.Insert 2.Display 3.Read 4.Exit

Read choice:1

Enter the value in the hash table:64

1.Insert 2.Display 3.Read 4.Exit

Read choice:2

Elements in the hash table are

At index 0 value = 0

At index 0 value = 0

At index 0 value = 12

At index 0 value = 24

At index 0 value = 44

At index 0 value = 64

At index 0 value = 0

At index 0 value = 67

At index 0 value = 58

At index 0 value = 0

1.Insert 2.Display 3.Read 4.Exit

Read choice:3

Enter the search element:44

Value is found at index 4

1.Insert 2.Display 3.Read 4.Exit

Read choice:4

CHAPTER –3

SAMPLE VIVA QUESTIONS

1. What is data structure?

The logical and mathematical model of a particular organization of data is called data structure. There are two types of data structure

1. Linear
2. Nonlinear

2. What are the goals of Data Structure?

- It must rich enough in structure to reflect the actual relationship of data in real world.
- The structure should be simple enough for efficient processing of data.

3. What does abstract Data Type Mean?

Data type is a collection of values and a set of operations on these values. Abstract data type refer to the mathematical concept that define the data type.It is a useful tool for specifying the logical properties of a data type.ADT consists of two parts

1. Values definition
2. Operation definition

4. What is the difference between a Stack and an Array?

- Stack is a ordered collection of items
- Stack is a dynamic object whose size is constantly changing as items are pushed and popped .
- Stack may contain different data types
- Stack is declared as a structure containing an array to hold the element of the stack, and an integer to indicate the current stack top within the array.

ARRAY

- Array is an ordered collection of items
- Array is a static object i.e. no of item is fixed and is assigned by the declaration of the array
- It contains same data types.
- Array can be home of a stack i.e. array can be declared large enough for maximum size of the stack.

5. What do you mean by recursive definition?

The definition which defines an object in terms of simpler cases of itself is called recursive definition.

6. What is sequential search?

In sequential search each item in the array is compared with the item being searched until a match occurs. It is applicable to a table organized either as an array or as a linked list.

7. What actions are performed when a function is called?

When a function is called

1. arguments are passed
2. local variables are allocated and initialized
3. transferring control to the function

8. What actions are performed when a function returns?

- Return address is retrieved
- Function's data area is freed
- Branch is taken to the return address

9. What is a linked list?

A linked list is a linear collection of data elements, called nodes, where the linear order is given by pointers. Each node has two parts first part contain the information of the element second part contains the address of the next node in the list.

10. What are the advantages of linked list over array (static data structure)?

The disadvantages of array are

- unlike linked list it is expensive to insert and delete elements in the array
- One can't double or triple the size of array as it occupies block of memory space.

In linked list

- each element in list contains a field, called a link or pointer which contains the address of the next element
- Successive element's need not occupy adjacent space in memory.

11. Can we apply binary search algorithm to a sorted linked list, why?

No we cannot apply binary search algorithm to a sorted linked list, since there is no way of indexing the middle element in the list. This is the drawback in using linked list as a data structure.

12. What do you mean by free pool?

Pool is a list consisting of unused memory cells which has its own pointer.

13. What do you mean by garbage collection?

It is a technique in which the operating system periodically collects all the deleted space onto the free storage list.It takes place when there is minimum amount of space left in storage list or when "CPU" is ideal.The alternate method to this is to immediately reinsert the space into free storage list which is time consuming.

14. What do you mean by overflow and underflow?

- When new data is to be inserted into the data structure but there is no available space i.e. free storage list is empty this situation is called overflow.
- When we want to delete data from a data structure that is empty this situation is called underflow.

15. What are the disadvantages array implementations of linked list?

1. The no of nodes needed can't be predicted when the program is written.
2. The no of nodes declared must remain allocated throughout its execution

16. What is a queue?

A queue is an ordered collection of items from which items may be deleted at one end (front end) and items inserted at the other end (rear end).It obeys FIFO rule there is no limit to the number of elements a queue contains.

17. What is a priority queue?

The priority queue is a data structure in which the intrinsic ordering of the elements (numeric or alphabetic) Determines the result of its basic operation. It is of two types

- Ascending priority queue- Here smallest item can be removed (insertion is arbitrary)
- Descending priority queue- Here largest item can be removed (insertion is arbitrary)

18. What are the disadvantages of sequential storage?

1.Fixed amount of storage remains allocated to the data structure even if it contains less element.

2.No more than fixed amount of storage is allocated causing overflow

19. What are the disadvantages of representing a stack or queue by a linked list?

1. A node in a linked list (info and next field) occupies more storage than a corresponding element in an array.
2. Additional time spent in managing the available list.

20. What is dangling pointer and how to avoid it?

After a call to free(p) makes a subsequent reference to *p illegal, i.e. though the storage to p is freed but the value of p(address) remain unchanged .so the object at that address may be used as the value of *p (i.e. there is no way to detect the illegality).Here p is called dangling pointer.

To avoid this it is better to set p to NULL after executing free(p).The null pointer value doesn't reference a storage location it is a pointer that doesn't point to anything.

21. What are the disadvantages of linear list?

1. We cannot reach any of the nodes that precede node (p)
2. If a list is traversed, the external pointer to the list must be persevered in order to reference the list again

22. Define circular list?

In linear list the next field of the last node contain a null pointer, when a next field in the last node contain a pointer back to the first node it is called circular list.

Advantages – From any point in the list it is possible to reach at any other point

23. What are the disadvantages of circular list?

1. We can't traverse the list backward
2. If a pointer to a node is given we cannot delete the node

24. Define double linked list?

It is a collection of data elements called nodes, where each node is divided into three parts

1. An info field that contains the information stored in the node
2. Left field that contain pointer to node on left side
3. Right field that contain pointer to node on right side

25. Is it necessary to sort a file before searching a particular item ?

If less work is involved in searching a element than to sort and then extract, then we don't go for sort

If frequent use of the file is required for the purpose of retrieving specific element, it is more efficient to sort the file.Thus it depends on situation.

26. What are the issues that hamper the efficiency in sorting a file?

The issues are

1. Length of time required by the programmer in coding a particular sorting program
2. Amount of machine time necessary for running the particular program
3. The amount of space necessary for the particular program .

27. Calculate the efficiency of sequential search?

The number of comparisons depends on where the record with the argument key appears in the table

1. If it appears at first position then one comparison
2. If it appears at last position then n comparisons
3. Average=(n+1)/2 comparisons
4. Unsuccessful search n comparisons
5. Number of comparisons in any case is O (n).

28. Is any implicit arguments are passed to a function when it is called?

Yes there is a set of implicit arguments that contain information necessary for the function to execute and return correctly. One of them is return address which is stored within the function's data area, at the time of returning to calling program the address is retrieved and the function branches to that location.

29. Parenthesis is never required in Postfix or Prefix expressions, why?

Parenthesis is not required because the order of the operators in the postfix /prefix expressions determines the actual order of operations in evaluating the expression

30. List out the areas in which data structures are applied extensively?

- Compiler Design,
- Operating System,
- Database Management System,
- Statistical analysis package,
- Numerical Analysis,
- Graphics,
- Artificial Intelligence,

- Simulation

31. What are the major data structures used in the following areas : network data model & Hierarchical data model?

RDBMS – Array (i.e. Array of structures)

Network data model – Graph

Hierarchical data model – Trees

32. If you are using C language to implement the heterogeneous linked list, what pointer type will you use?

The heterogeneous linked list contains different data types in its nodes and we need a link, pointer to connect them. It is not possible to use ordinary pointers for this. So we go for void pointer. Void pointer is capable of storing pointer to any type as it is a generic pointer type.

33. Minimum number of queues needed to implement the priority queue?

Two. One queue is used for actual storing of data and another for storing priorities.

34. What is the data structures used to perform recursion?

Stack. Because of its LIFO (Last In First Out) property it remembers its 'caller' so knows whom to return when the function has to return. Recursion makes use of system stack for storing the return addresses of the function calls. Every recursive function has its equivalent iterative (non-recursive) function. Even when such equivalent iterative procedures are written, explicit stack is to be used.

35. What are the notations used in Evaluation of Arithmetic Expressions using prefix and postfix forms?

Polish and Reverse Polish notations.

36. Convert the expression ((A + B) * C – (D – E) ^ (F + G)) to equivalent Prefix and Postfix notations?

1.Prefix Notation:

^ – * +ABC – DE + FG

2.Postfix Notation:

AB + C * DE – – FG + ^

37. Sorting is not possible by using which of the following methods?

(a) Insertion

(b) Selection

(c) Exchange

(d) Deletion

(d) Deletion.

Using insertion we can perform insertion sort, using selection we can perform selection sort, using exchange we can perform the bubble sort (and other similar sorting methods). But no sorting method can be done just using deletion.

38. List out few of the Application of tree data-structure?

The manipulation of Arithmetic expression,

Symbol Table construction,

Syntax analysis.

39. List out few of the applications that make use of Multilinked Structures?

Sparse matrix, Index generation.

40. in tree construction which is the suitable efficient data structure?

(A) Array (b) Linked list (c) Stack (d) Queue (e) none

(b) Linked list

41. What is the type of the algorithm used in solving the 8 Queens problem?

Backtracking

42. In an AVL tree, at what condition the balancing is to be done?

If the 'pivotal value' (or the 'Height factor') is greater than 1 or less than –1.

43. In RDBMS, what is the efficient data structure used in the internal storage representation?

B+ tree. Because in B+ tree, all the data is stored only in leaf nodes, that makes searching easier. This corresponds to the records that shall be stored in leaf nodes.

45. One of the following tree structures, which is, efficient considering space and time complexities?

a) Incomplete Binary Tree.

b) Complete Binary Tree.

c) Full Binary Tree.

b) Complete Binary Tree.

By the method of elimination:

Full binary tree loses its nature when operations of insertions and deletions are done. For incomplete binary trees, extra property of complete binary tree is maintained even after operations like additions and deletions are done on it.

46. What is a spanning Tree?

A spanning tree is a tree associated with a network. All the nodes of the graph appear on the tree once. A minimum spanning tree is a spanning tree organized so that the total edge weight between nodes is minimized.

47. Does the minimum spanning tree of a graph give the shortest distance between any 2 specified nodes?

No.Minimal spanning tree assures that the total weight of the tree is kept at its minimum. But it doesn't mean that the distance between any two nodes involved in the minimum-spanning tree is minimum.

48. Whether Linked List is linear or Non-linear data structure?

According to Storage Linked List is a Non-linear one.

CHAPTER-4

SAMPLE PROJECTS

1. Snake Game

Snake Game develops a classic Snake Game where the snake grows longer with every food item it eats, avoiding collisions with the wall or itself. It's one of the most fun and interactive DSA project ideas for beginners.

- **Key Concepts Used:** Arrays, Game Loop, Collision Detection
- **Best For:** DSA projects with source code in Python or Java

2. Sorting Visualizer

The Sorting Visualizer creates an interactive tool to visualize how sorting algorithms work in real-time. It's great for building a deeper understanding of algorithm efficiency and visual representation.

- **Key Concepts Used:** Bubble Sort, Merge Sort, Quick Sort
- **Best For:** Web-based DSA projects in JavaScript or DSA projects in C++

3. Maze Solver

Maze Solve can solve mazes using BFS or DFS. This is a perfect DSA-based project for learners looking to master graph traversal and backtracking.

- **Key Concepts Used:** Graphs, BFS, DFS, Backtracking
- **Ideal For:** Intermediate-level DSA projects in Java or Python

4. Linked List Implementation

Build a complete implementation of singly and doubly linked lists from scratch. This foundational project is essential for understanding dynamic data structures.

- **Key Concepts Used:** Pointers, Memory Allocation, Linked List Operations
- **Best For:** Fundamental DSA projects in C++ or Java

Explore More: Mini Project Ideas for Computer Science Students | MERN Stack Project Ideas

5. Binary Tree Construction

Construct binary trees with traversal operations. A key DSA project for resume building, showcasing recursion and tree algorithms.

- **Key Concepts Used:** Pre-order, In-order, Post-order Traversal, Recursion
- **Great For:** DSA projects in Java and Python

6. Graph Algorithms Implementation

Implement graph algorithms like Dijkstra's and DFS to solve real-world problems. This is one of the most powerful DSA projects with source code for learning advanced concepts.

- **Key Concepts Used:** Graph Traversals, Dijkstra's, Cycle Detection
- **Best In:** DSA projects in C++ or Python

Also Read: **Types of Graphs in Data Structures & Applications**

7. Sudoku Solver

Create a solver that uses backtracking to solve Sudoku puzzles. A fun and challenging DSA-based project to practice recursion and constraint satisfaction.

- **Key Concepts Used:** Recursion, Backtracking, Constraint Satisfaction
- **Recommended For:** Python or Java implementations

8. Travel Planner using Graphs

Design a system that plans the shortest travel routes using Dijkstra's algorithm. It's a great example of applying graph algorithms in real-world scenarios.

- **Key Concepts Used:** Weighted Graphs, Dijkstra's Algorithm
- **Perfect For:** DSA projects for a resume in logistics or transport applications

9. File Zipper Project

Build a tool that compresses and extracts files using Huffman Coding. It's a great DSA project idea that teaches you about binary trees and compression techniques.

- **Key Concepts Used:** Huffman Coding, File I/O, Bit Manipulation
- **Best For:** DSA projects in C++ or Python

10. Dynamic Event Scheduling Using Graph

Create a system to efficiently schedule events by avoiding overlaps using graph coloring. It's a smart project that blends graphs with real-world planning.

- **Key Concepts Used:** Graph Coloring, Greedy Algorithms
- **Great For:** Advanced DSA projects in Java or Python

11. Social Media Trend Analyzer Using Trie and Heap

Using Trie and Heap, Track and analyze trending hashtags using Trie for storage and Heap for ranking. A real-time, advanced DSA project that deals with large-scale data.

- **Key Concepts Used:** Trie, Heap, Real-time Data Processing
- **Ideal For:** DSA projects in Java and Python

This project applies to social media analytics, sentiment analysis, and recommendation systems, especially for businesses tracking online trends and customer preferences.

12. Creating a To-Do List

Build a simple to-do list app using basic data structures. This beginner-friendly DSA project idea helps you understand CRUD operations and UI handling.

- **Key Concepts Used:** Arrays, Lists, CRUD, User Interaction
- **Suggested In:** Python or JavaScript

13. Building a Phonebook

Implement a contact management system using hashing for fast lookup. It's a solid project for showcasing your skills in efficient data retrieval.

- **Key Concepts Used:** Hash Maps, Search Algorithms, Data Management
- **Best For:** DSA projects with source code in Java or C++

14. Build a Simple Calculator

Design a calculator to perform basic arithmetic functions with UI support. A useful beginner project to understand event handling and user input.

- **Key Concepts Used:** Arithmetic Logic, Input Validation, UI Handling

- **Recommended In:** Python with Tkinter or Java with JavaFX

Also, for a fun practice, explore Build a Calculator using JavaScript, HTML, and CSS in 2025!

15. Students' Grade Checker

Create a tool to input student marks, compute grades, and track performance. This is a basic yet important data project idea for managing academic data.

- **Key Concepts Used:** Conditional Logic, File I/O, Data Display
- **Ideal For:** Python or Java-based project

16. Plagiarism Detection System

Detect content similarity between texts using string-matching algorithms. A great project to apply text analysis and algorithmic thinking.

- **Key Concepts Used:** String Matching, Text Analysis, Recursion
- **Best For:** DSA projects in Java or Python

17. Crossword Puzzle Game

Build a crossword puzzle using backtracking and word constraints. A fun way to explore 2D arrays and interactive puzzle design.

- **Key Concepts Used:** Backtracking, Constraint Satisfaction, 2D Arrays
- **Great For:** Beginner-to-intermediate level DSA projects.

18. Task Scheduler

Develop a scheduler that sorts and prioritizes tasks using heaps. This DSA project is useful in time management and productivity tools.

- **Key Concepts Used:** Priority Queue, Heap, Dynamic Data Management
- **Recommended In:** Python or Java

19. Pathfinding Algorithms Visualizer

Create a visual tool to demonstrate how pathfinding algorithms work in grids. It’s a must-have DSA project for a resume that displays algorithmic thinking clearly.

- **Key Concepts Used:** A*, Dijkstra's, BFS, Visualization
- **Perfect For:** DSA projects in JavaScript or Python

20. Library Management System

Build a tool to manage books, members, and borrowing activities like a real-world library. This DSA project is great for practicing data storage and user management.

- **Key Concepts Used:** Linked List, Hash Map, Queues, File Handling
- **Recommended In:** DSA projects in Java or C++

21. Social Network Analysis

Use graph algorithms to analyze user connections and trends in a social media-style network. It's one of the best DSA project ideas to explore clustering and shortest paths.

- **Key Concepts Used:** Graph Traversal, Clustering, Community Detection
- **Recommended In:** Python with NetworkX or DSA projects in Java

22. Banking Management System

Simulate bank operations like deposits, withdrawals, and balance checks. This project teaches secure transaction processing and real-time data handling.

- **Key Concepts Used:** Binary Search Trees, Queues, Security Checks
- **Recommended In:** DSA projects in Java or C++

4. Travel Planner using Graphs

Design a travel planner that finds the best route based on cost, distance, or time. This is a top pick among DSA projects involving graph theory.

- **Key Concepts Used:** Dijkstra's Algorithm, Graphs, Shortest Path
- **Recommended In:** Python with NetworkX or JavaScript with Google Maps API

23. Cash Flow Minimizer

Create a system to manage and optimize financial transactions. A smart DSA project for resume building in the finance domain.

- **Key Concepts Used:** Greedy Algorithms, Dynamic Programming
- **Recommended In:** Python or C++ for algorithm-heavy tasks

24. E-commerce Inventory Management System

Build an app that tracks products, handles orders, and manages stock in real-time. Perfect for learning how to apply DSA in retail systems.

- **Key Concepts Used:** Hashing, Search Trees, Real-time Updates
- **Recommended In:** DSA projects in Java or Python

25. Job Scheduling Algorithm

Design a system that schedules jobs based on deadlines or profits. A strong DSA project idea for learning optimization techniques.

- **Key Concepts Used:** Greedy Algorithms, Job Prioritization
- **Recommended In:** Python or Java with standard libraries

26. Real-Time Stock Price Analysis

Track and analyze livestock data to predict market trends. Great for combining DSA knowledge with real-time APIs and financial data.

- **Key Concepts Used:** Moving Averages, APIs, Data Visualization
- **Recommended In:** DSA projects in Python or JavaScript

27. Movie Recommendation System

Build a system that suggests movies to users using collaborative filtering techniques. It’s a smart DSA project that shows practical use of matrix operations and similarity scoring.

- **Key Concepts Used:** Collaborative Filtering, User-Item Matrix, Similarity Metrics
- **Recommended In:** Python or DSA projects in Java using Scikit-learn

28. URL Shortener Service

Create a tool that shortens long URLs into unique short links using hashing and redirection. This is one of the most practical DSA projects used in web applications.

- **Key Concepts Used:** Hashing, Redirection, Database Mapping
- **Recommended In:** Python with Flask or Node.js using Express

29. Data Compression with Huffman Encoding

Develop a file compression system using Huffman Coding. This project is perfect for learning greedy algorithms and binary trees, commonly covered in DSA projects in C++ and Java.

- **Key Concepts Used:** Huffman Trees, Greedy Algorithms, Bit Manipulation
- **Recommended In:** DSA projects in C++ or Java with custom implementation

30. Predictive Text Input Using Trie

Build an autocomplete tool that predicts words as a user types. This project is excellent for mastering the Trie data structure and efficient string searching.

- **Key Concepts Used:** Trie, String Matching, Real-time Search
- **Recommended In:** DSA projects in Java or Python with custom implementation

31. Graph-Based Sudoku Solver

Solve Sudoku puzzles using graph theory by treating each cell as a node and constraints as edges. A unique DSA project idea for those who enjoy logic-heavy challenges.

- **Key Concepts Used:** Graphs, Backtracking, Constraint Propagation
- **Recommended In:** Python with Pygame or Java with GUI

32. Stock Price Prediction

Use historical data and machine learning models to predict future stock prices. This project blends DSA with financial forecasting and real-time analysis.

- **Key Concepts Used:** Time Series Analysis, ML Algorithms, Data Visualization
- **Recommended In:** Python with Scikit-learn, Pandas, and TensorFlow

33. Chatbot with Real-Time Sentiment Analysis

Build an intelligent chatbot that understands user queries and responds in real-time using NLP and sentiment detection. A high-impact DSA project for a resume in AI-focused roles.

- **Key Concepts Used:** NLP, Machine Learning, Real-time Interaction
- **Recommended In:** Python with NLTK, SpaCy, and TensorFlow