||Jai Sri Gurudev||

Sri AdichunchanagiriShikshana Trust(R)

SJB Institute of Technology

No. 67, BGS Health & Education City, Dr. Vishnuvardhan Road Kengeri, Bangalore – 560 060

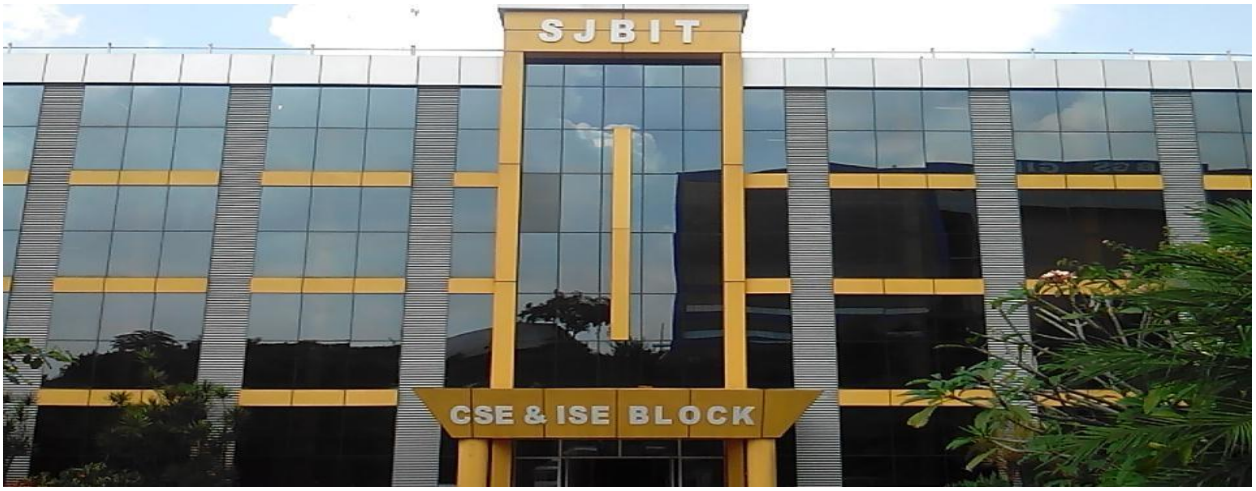
SJBIT

Department of Computer Science and Engineering

Data Structures and Applications Laboratory

[BCSL305]

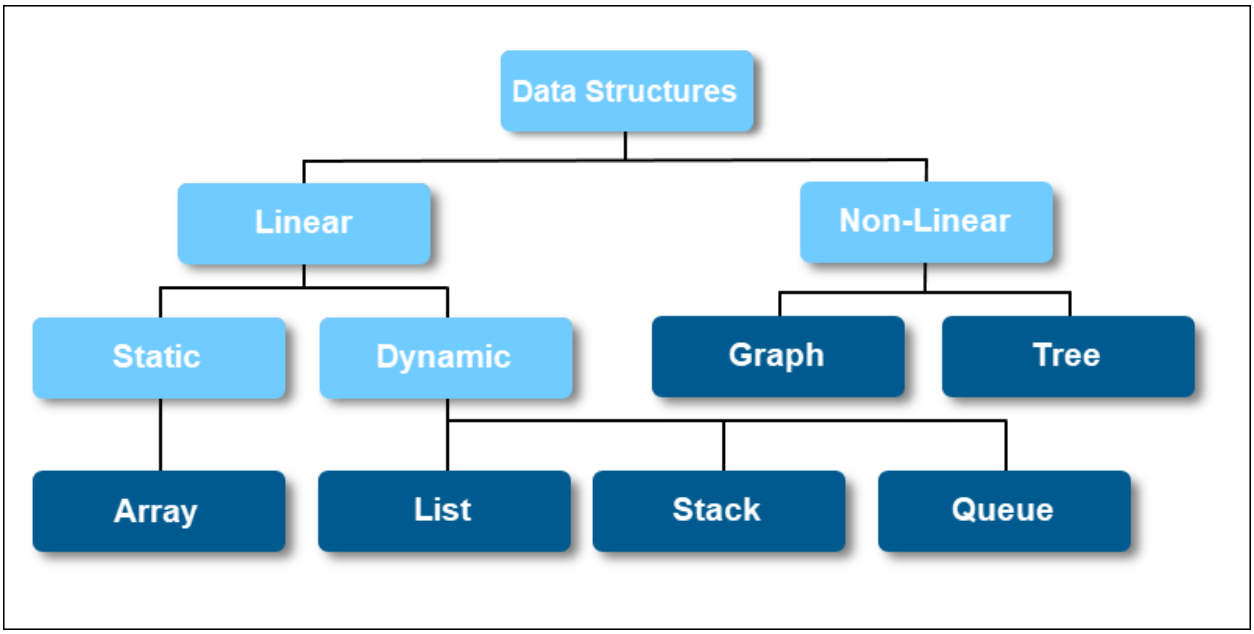
III SEMESTER – B. E



Staff Name	**Dr.shanth kumar, Dr.Ajay B N, Arun Kumar D R**		
Section	**A, B, C**	**Batch**	

PREFACE

A data structure a way of arranging data on a computer so that it can be accessed and updated efficiently. A data structure is not only used for organizing the data. It is also used for processing, retrieving, and storing data. There are different basic and advanced types of data structures that are used in almost every program or software system that has been developed.



1. **Linear data structure:** Data structure in which data elements are arranged sequentially or linearly, where each element is attached to its previous and next adjacent elements, is called a linear data structure. Examples of linear data structures are array, stack, queue, linked list, etc.
   - **Static data structure:** Static data structure has a fixed memory size. It is easier to access the elements in a static data structure. An example of this data structure is an array.
   - **Dynamic data structure:** In dynamic data structure, the size is not fixed. It can be randomly updated during the runtime which may be considered efficient concerning the memory (space) complexity of the code. Examples of this data structure are queue, stack, etc.
2. **Non-linear data structure:** Data structures where data elements are not placed sequentially or linearly are called non-linear data structures. In a non-linear data structure, we can't traverse all the elements in a single run only. Examples of non-linear data structures are trees and graphs.

SJB INTITUTE OF TECHNOLOGY

Institution's Vision

To become a recognized technical education centre with a global perspective.

Institution's Mission

To provide learning opportunities that foster students ethical values, intelligent development in science & technology and social responsibility so that they become sensible and contributing members of the society.

Department of Information Science and Engineering

Department Vision

We envision our department as a catalyst for developing educated, engaged and employable individuals whose collective energy will be the driving force for prosperity and quality of life in our diverse world.

Department Mission

To establish Our mission is to provide quality technical education in the field of information technology and to strive for excellence in the education by developing and sharpening the intellectual and human potential for good industry and community.

Program Educational Objectives (PEO's)

Graduates will -

PEO1: Possess expertise in problem solving, design and analysis, technical skills for a fruitful carrier accomplishing professional and social ethics with exposure to modern designing tools and technologies in information science and engineering.

PEO2: excel in communication, teamwork, and multiple domains related to engineering issues accomplishing social responsibilities and management skills.

PEO3: Acquire outclass in competitive environment through certification courses, gaining leadership qualities and progressive research to become successful entrepreneurs.

Program Specific Outcomes (PSO'S)

Graduates will be able to -

PSO1: Analyze, Apply knowledge of information science to develop software solutions in current research trends and technology.

PSO2: Develop analytical and experimental skills in the field of Transportation, Water Resources, Environmental Engineering, Construction Technology and Project management.

Program Outcomes-POs

Engineering graduates will be able to:

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

CONTENT

	a. Create a SLL of N Students Data by using *front insertion*. b. Display the status of SLL and count the number of nodes in it c. Perform Insertion / Deletion at End of SLL d. Perform Insertion / Deletion at Front of SLL(Demonstration of stack) e. Exit	
8	Develop a menu driven Program in C for the following operations on Doubly Linked List (DLL) of Employee Data with the fields: *SSN, Name, Dept, Designation, Sal, PhNo* a. Create a DLL of N Employees Data by using *end insertion*. b. Display the status of DLL and count the number of nodes in it c. Perform Insertion and Deletion at End of DLL d. Perform Insertion and Deletion at Front of DLL e. Demonstrate how this DLL can be used as Double Ended Queue. f. Exit	27
9	Develop a Program in C for the following operationson Singly Circular Linked List (SCLL) with header nodes a. Represent and Evaluate a Polynomial $P(x,y,z) = 6x^2y^2z-4yz^5+3x^3yz+2xy^5z-2xyz^3$ b. Find the sum of two polynomials POLY1(x,y,z) and POLY2(x,y,z) and store the result in POLYSUM(x,y,z) Support the program with appropriate functions for each of the above operations	33
10	Develop a menu driven Program in C for the following operations on Binary Search Tree (BST) of Integers . a. Create a BST of N Integers: 6, 9, 5, 2, 8, 15, 24, 14, 7, 8, 5, 2 b. Traverse the BST in Inorder, Preorder and Post Order c. Search the BST for a given element (KEY) and report the appropriate message d. Exit	40
11	Develop a Program in C for the following operations on Graph(G) of Cities a. Create a Graph of N cities using Adjacency Matrix. b. Print all the nodes reachable from a given starting node in a digraph using DFS/BFS method	44
12	Given a File of N employee records with a set K of Keys (4-digit) which uniquely determine the records in file F. Assume that file F is maintained in memory by a Hash Table (HT) of m memory locations with L as the set of memory addresses (2-digit) of locations in HT. Let the keys in K and addresses in L are Integers. Develop a Program in C that uses Hash function H: K →L as H(K)=K mod m (remainder method), and implement hashing technique to map a given key K to the address space L. Resolve the collision (if any) using linear probing.	47

Course Learning Objectives:

This laboratory course enables students to get practical experience in design, develop, implement, analyze and evaluation/testing of

- Dynamic memory management
- Linear data structures and their applications such as stacks, queues and lists
- Non-Linear data structures and their applications such as trees and graphs

Course outcome (Course Skill Set)

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

- Analyze various linear and non-linear data structures
- Demonstrate the working nature of different types of data structures and their applications
- Use appropriate searching and sorting algorithms for the give scenario.
- Apply the appropriate data structure for solving real world problems

1. Develop a Program in C for the following:
a) Declare a calendar as an array of 7 elements (A dynamically Created array) to represent 7 days of a week. Each Element of the array is a structure having three fields. The first field is the name of the Day (A dynamically allocated String), The second field is the date of the Day (A integer), the third field is the description of the activity for a
particular day (A dynamically allocated String).
b) Write functions create(), read() and display(); to create the calendar, to read the data from the keyboard and to print weeks activity details report on screen.

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

struct Day {
   char* name;
   int date;
   char* activity;
};

struct Day* calendar[7];

void create() {
   for (int i = 0; i < 7; i++) {
      calendar[i] = (struct Day*)malloc(sizeof(struct Day));
      calendar[i]->name = (char*)malloc(20 * sizeof(char)); // Assuming the maximum length of the day name is 20
      calendar[i]->activity = (char*)malloc(100 * sizeof(char)); // Assuming the maximum length of the activity description is 100

      printf("Enter day name: ");
      scanf("%s", calendar[i]->name);
      printf("Enter date: ");
      scanf("%d", &calendar[i]->date);
      printf("Enter activity: ");
      scanf(" %[^\n]",calendar[i]->activity);
      printf("\n");
   }
}

void read() {
   for (int i = 0; i < 7; i++) {
      printf("Day %d\n", i + 1);
      printf("Day name: %s\n", calendar[i]->name);
      printf("Date: %d\n", calendar[i]->date);
      printf("Activity: %s\n", calendar[i]->activity);
      printf("\n");
   }
}

void display() {
```

```
    printf("Weekly Activity Details:\n");
    for (int i = 0; i < 7; i++) {
        printf("Day %d: %s - %d, Activity: %s\n", i + 1, calendar[i]->name, calendar[i]->date, calendar[i]->activity);
    }
}

int main() {
    printf("Creating Calendar:\n");
    create();

    printf("Reading Calendar:\n");
    read();

    printf("Displaying Calendar:\n");
    display();

    // Free memory
    for (int i = 0; i < 7; i++) {
        free(calendar[i]->name);
        free(calendar[i]->activity);
        free(calendar[i]);
    }

    return 0;
}
```

Output:

```
Creating Calendar:
Enter day name: Sunday
Enter date: 14
Enter activity: Sports

Enter day name: Monday
Enter date: 15
Enter activity: Data structures lab

Enter day name: Tuesday
Enter date: 16
Enter activity: Analog lab

Enter day name: Wednesday
Enter date: 17
Enter activity: Theory classes

Enter day name: Thursday
Enter date: 18
Enter activity: Project work
```

Enter day name: Friday
Enter date: 19
Enter activity: Internship

Enter day name: Saturday
Enter date: 20
Enter activity: Social connect

Reading Calendar:
Day 1
Day name: Sunday
Date: 14
Activity: Sports

Day 2
Day name: Monday
Date: 15
Activity: Data structures lab

Day 3
Day name: Tuesday
Date: 16
Activity: Analog lab

Day 4
Day name: Wednesday
Date: 17
Activity: Theory classes

Day 5
Day name: Thursday
Date: 18
Activity: Project work

Day 6
Day name: Friday
Date: 19
Activity: Internship

Day 7
Day name: Saturday
Date: 20
Activity: Social connect

Displaying Calendar:
Weekly Activity Details:
Day 1: Sunday - 14, Activity: Sports
Day 2: Monday - 15, Activity: Data structures lab
Day 3: Tuesday - 16, Activity: Analog lab
Day 4: Wednesday - 17, Activity: Theory classes

Day 5: Thursday - 18, Activity: Project work
Day 6: Friday - 19, Activity: Internship
Day 7: Saturday - 20, Activity: Social connect

2. Develop a Program in C for the following operations on Strings.
a. Read a main String (STR), a Pattern String (PAT) and a Replace String (REP)
b. Perform Pattern Matching Operation: Find and Replace all occurrences of PAT in STR with REP if PAT exists in STR. Report suitable messages in case PAT does not exist in STR Support the program with functions for each of the above operations. Don't use Built-in functions.

```
#include<stdio.h>
char str[50], pat[20], rep[20], ans[50];
int     c=0, m=0, i=0, j=0, k, flag=0;
void stringmatch()
{
            while(str[c] !='\0')
            {
                        if(str[m] == pat[i])
                        {
                                    i++;
                                    m++;
                                    if(pat[i] == '\0')
                                    {
                                                flag = 1;
                                                for(k=0; rep[k]!='\0'; k++, j++)
                                                {
                                                            ans[j] = rep[k];
                                                }
                                                i = 0;
                                                c = m;
                                    }
                        }
                        else
                        {

                                    ans[j]= str[c];
                                    j++;
                                    c++;
                                    m=c;
                                    i=0;
                        }
            }
            ans[j]='\0';
}
void main()
{
            printf("\nEnter the main string:");
```

```
gets(str);
```

```
        printf("\nEnter the pat string:");
        gets(pat);
        printf("\nEnter the replace string:");
        gets(rep);
        stringmatch();
        if(flag == 1)
                printf("\nResultant string is: %s", ans);
        else
                printf("\nPattern string is not found");
}
```

Output:

Enter the main string:I love apples. An apple a day keeps the doctor away

Enter the pat string:apple

Enter the replace string:orange

Resultant string is: I love oranges. An orange a day keeps the doctor away

3. 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 how Stack can be used to check Palindrome
d. Demonstrate Overflow and Underflow situations on Stack
e. Display the status of Stack
f. Exit
Support the program with appropriate functions for each of the above operations.

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 5

int s[MAX];
int top = -1;

void push(int item);
int pop();
void palindrome();
void display();
void main()
{
        int choice, item;
        while(1)
        {
```

```
                printf("\n\n\n\n~~~~~~Menu~~~~~~ : ");
                printf("\n=>1.Push an Element to Stack and Overflow demo ");
                printf("\n=>2.Pop an Element from Stack and Underflow demo");
                printf("\n=>3.Palindrome demo ");
                printf("\n=>4.Display ");
                printf("\n=>5.Exit");
                printf("\nEnter your choice: ");
                scanf("%d", &choice);
                switch(choice)
                {
                        case 1:         printf("\nEnter an element to be pushed: ");
                                        scanf("%d", &item);


                                        push(item);
                                        break;
                        case 2:         item = pop();
                                        if(item != -1)
                                                printf("\nElement popped is: %d", item);
                                        break;
                        case 3:         palindrome();
                                        break;
                        case 4:         display();
                                        break;
                        case 5:         exit(1);
                        default:        printf("\nPlease enter valid choice ") ;
                                        break;
                }
        }
}

void push(int item)
{
        if(top == MAX-1)
        {
                printf("\n~~~~Stack overflow~~~~");
                return;
        }

        top = top + 1 ;
        s[top] = item;
}

int pop()
{
        int item;
        if(top == -1)
        {
                printf("\n~~~~Stack underflow~~~~");
                return -1;
```

```
        }
        item = s[top];
        top = top - 1;
        return item;
}

void display()
{
        int i;
        if(top == -1)
        {
                printf("\n~~~~Stack is empty~~~~");
                return;
        }
        printf("\nStack elements are:\n ");
        for(i=top; i>=0 ; i--)
                printf("| %d |\n", s[i]);
}

void palindrome()
{
        int flag=1,i;
        printf("\nStack content are:\n");
        for(i=top; i>=0 ; i--)
                printf("| %d |\n", s[i]);

        printf("\nReverse of stack content are:\n");
        for(i=0; i<=top; i++)
                printf("| %d |\n", s[i]);

        for(i=0; i<=top/2; i++)
        {
                if( s[i] != s[top-i] )
                {
                        flag = 0;
                        break;
                }
        }
        if(flag == 1)
        {
                printf("\nIt is palindrome number");
        }
        else
        {
                printf("\nIt is not a palindrome number");
        }
}
```

Output:

```
~~~~~~Menu~~~~~~ :
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: 1

Enter an element to be pushed: 3

~~~~~~Menu~~~~~~ :
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: 1

Enter an element to be pushed: 4

~~~~~~Menu~~~~~~ :
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: 1

Enter an element to be pushed: 3
~~~~~~Menu~~~~~~ :
=>1.Push an Element to Stack and Overflow demo
=>2.Pop an Element from Stack and Underflow demo
=>3.Palindrome demo
=>4.Display
=>5.Exit
Enter your choice: 3

Stack content are:
| 3 |
| 4 |
| 3 |

Reverse of stack content are:
| 3 |
| 4 |
| 3 |

It is palindrome number
```

4. 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>

void  evaluate();
void push(char);
char pop();
int prec(char);

char infix[30], postfix[30], stack[30];
int top = -1;

void main()
{
        printf("\nEnter the valid infix expression:\t");
        scanf("%s", infix);
        evaluate();
        printf("\nThe entered infix expression is :\n %s \n", infix);
        printf("\nThe corresponding postfix expression is :\n %s \n", postfix);
}

void evaluate()
{
        int i = 0, j = 0;
        char symb, temp;

        push('#');

        for(i=0; infix[i] != '\0'; i++)
        {
                symb = infix[i];
                switch(symb)
                {
                        case '(' :      push(symb);
                                        break;

                        case ')' :      temp = pop();
                                        while(temp != '(' )
                                        {
                                                postfix[j] = temp;
                                                j++;
                                                temp = pop();
                                        }
                                        break;
                        case '+' :
```

```
                        case '-' :
                        case '*' :
                        case '/' :
                        case '%' :
                        case '^' :
                        case '$'  :     while( prec(stack[top]) >= prec(symb) )
                                        {
                                                temp = pop();
                                                postfix[j] = temp;
                                                j++;
                                        }
                                        push(symb);
                                        break;
                        default:        postfix[j] = symb;
                                        j++;
                }
        }
        while(top > 0)
        {
                temp = pop();
                postfix[j] = temp;
                j++;
        }
        postfix[j] = '\0';
}

 void push(char item)
{
        top = top+1;
        stack[top] = item;
}

 char pop()
{
        char item;
        item = stack[top];
        top = top-1;
        return item;
}

int prec(char symb)
{
        int p;
        switch(symb)
        {
                case '#' :      p = -1;
                                break;

                case '(' :
                case ')' :      p = 0;
```

```
                        break;

                case '+' :
                case '-' :      p = 1;
                                break;

                case '*' :
                case '/' :
                case '%' :      p = 2;
                                break;

                case '^' :
                case '$' :      p = 3;
                                break;
        }
        return p;
}
```

Output:

```
Enter the valid infix expression:       (a+b)+c/d*e

The entered infix expression is :
 (a+b)+c/d*e

The corresponding postfix expression is :
 ab+cd/e*+
```

5. Develop a Program in C for the following Stack Applications
a. Evaluation of Suffix expression with single digit operands and operators: +, -, *, /, %, ^
b. Solving Tower of Hanoi problem with n disks

Program 5a

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

int i, top = -1;
int op1, op2, res, s[20];
char postfix[90], symb;

void push(int item)
{
        top = top+1;
        s[top] = item;
}

int pop()
{
```

```
        int item;
        item = s[top];
        top = top-1;
        return item;
}

void main()
{
        printf("\nEnter a valid postfix expression:\n");
        scanf("%s", postfix);
        for(i=0; postfix[i]!='\0'; i++)
        {
                symb = postfix[i];
                if(isdigit(symb))
                {
                        push(symb - '0');
                }
                else
                {
                        op2  =  pop();
                        op1  =  pop();
                        switch(symb)
                        {
                                case '+':       push(op1+op2);
                                                break;
                                case '-':       push(op1-op2);
                                                break;
                                case '*':       push(op1*op2);
                                                break;
                                case '/':       push(op1/op2);
                                                break;
                                case '%':       push(op1%op2);
                                                break;
                                case '$':
                                case '^':       push(pow(op1, op2));
                                                break;
                                default :  push(0);
                        }
                }
        }
        res = pop();
        printf("\n Result = %d", res);
}
```

Output:

```
Enter a valid postfix expression:
623+-382/+*2$3+
Result = 52
```

```
Enter a valid postfix expression:
42$3*3-84/11+/+
Result = 46
```

Program 5b:

```
#include<stdio.h>
#include<math.h>
void tower(int n, char from_peg,  char aux_peg, char to_peg);

void main()
{
        int n;
        printf("\nEnter the number of disks: ");
        scanf("%d", &n);
        tower(n, 'A', 'B', 'C');                        // A-> from_peg B->aux_peg C->to_peg
   printf("\nTotal number of moves = %0.0f", pow(2,n)-1 );
}

void tower(int n, char from_peg,  char aux_peg, char to_peg)
// A-> from_peg B->aux_peg C->to_peg
{
        if(n == 1)
        {
                printf("\nMove disk %d from %c peg to %c peg", n, from_peg, to_peg);
                return;
        }


        // move n-1 disks from A(from_peg) to B(to_peg) using C(aux_peg) as auxiliary
        tower(n-1, from_peg,  to_peg,  aux_peg);

        printf("\nMove disk %d from peg %c to %c peg", n, from_peg,  to_peg);

        // move n-1 disks from B(aux_peg) to C(to_peg) using A(from_peg) as auxiliary
        tower(n-1, aux_peg, from_peg, to_peg);
}
```

Output:

```
Enter the number of disks: 3
Move disk 1 from A peg to C peg
Move disk 2 from peg A to B peg
Move disk 1 from C peg to B peg
Move disk 3 from peg A to C peg
Move disk 1 from B peg to A peg
Move disk 2 from peg B to C peg
Move disk 1 from A peg to C peg
Total number of moves = 7
```

6. Develop a menu driven Program in C for the following operations on Circular QUEUE of Characters (Array Implementation of Queue with maximum size MAX)
a. Insert an Element on to Circular QUEUE
b. Delete an Element from Circular QUEUE
c. Demonstrate Overflow and Underflow situations on Circular QUEUE
d. Display the status of Circular QUEUE
e. Exit
Support the program with appropriate functions for each of the above operations

```
// Circular Queue implementation in C

#include <stdio.h>

#define SIZE 5

int items[SIZE];
int front = -1, rear = -1;

// Check if the queue is full
int isFull() {
  if ((front == rear + 1) || (front == 0 && rear == SIZE - 1)) return 1;
  return 0;
}

// Check if the queue is empty
int isEmpty() {
  if (front == -1) return 1;
  return 0;
}

// Adding an element
void enQueue(int element) {
  if (isFull())
    printf("\n Queue is full!! \n");
  else {
    if (front == -1) front = 0;
    rear = (rear + 1) % SIZE;
    items[rear] = element;
    printf("\n Inserted -> %d", element);
  }
}

// Removing an element
int deQueue() {
  int element;
  if (isEmpty()) {
    printf("\n Queue is empty !! \n");
    return (-1);
  } else {
    element = items[front];
```

```
    if (front == rear) {
      front = -1;
      rear = -1;
    }
    // Q has only one element, so we reset the
    // queue after dequeing it. ?
    else {
      front = (front + 1) % SIZE;
    }
    printf("\n Deleted element -> %d \n", element);
    return (element);
  }
}

// Display the queue
void display() {
  int i;
  if (isEmpty())
    printf(" \n Empty Queue\n");
  else {
    printf("\n Front -> %d ", front);
    printf("\n Items -> ");
    for (i = front; i != rear; i = (i + 1) % SIZE) {
      printf("%d ", items[i]);
    }
    printf("%d ", items[i]);
    printf("\n Rear -> %d \n", rear);
  }
}

int main()
{
    int choice=1,x;  // variables declaration

    while(choice<4 && choice!=0)  // while loop
    {
    printf("\n Press 1: Insert an element");
    printf("\nPress 2: Delete an element");
    printf("\nPress 3: Display the element");
    printf("\nEnter your choice");
    scanf("%d", &choice);

    switch(choice)
    {

        case 1:

        printf("Enter the element which is to be inserted");
        scanf("%d", &x);
        enQueue(x);
```

```
        break;
        case 2:
        deQueue();
        break;
        case 3:
        display();

    }}
    return 0;
}
```

Output:

```
 Press 1: Insert an element
Press 2: Delete an element
Press 3: Display the element
Enter your choice1
Enter the element which is to be inserted11

 Inserted -> 11
 Press 1: Insert an element
Press 2: Delete an element
Press 3: Display the element
Enter your choice1
Enter the element which is to be inserted22

 Inserted -> 22
 Press 1: Insert an element
Press 2: Delete an element
Press 3: Display the element
Enter your choice1
Enter the element which is to be inserted33

 Inserted -> 33
 Press 1: Insert an element
Press 2: Delete an element
Press 3: Display the element
Enter your choice3

 Front -> 0
 Items -> 11 22 33
 Rear -> 2

 Press 1: Insert an element
Press 2: Delete an element
Press 3: Display the element
Enter your choice2

 Deleted element -> 11
```

```
 Press 1: Insert an element
Press 2: Delete an element
Press 3: Display the element
Enter your choice3

 Front -> 1
 Items -> 22 33
 Rear -> 2
```

7. Develop a menu driven Program in C for the following operations on Singly Linked List (SLL) of Student Data with the fields: USN, Name, Programme, Sem, PhNo
a. Create a SLL of N Students Data by using front insertion.
b. Display the status of SLL and count the number of nodes in it
c. Perform Insertion / Deletion at End of SLL
d. Perform Insertion / Deletion at Front of SLL(Demonstration of stack)
e. Exit

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

struct node
{
   char usn[25],name[25],programme[25];
   int sem;
   long int phone;
   struct node *link;
};
typedef struct node * NODE;

NODE start = NULL;
int count=0;


NODE create()
{
   NODE snode;
   snode = (NODE)malloc(sizeof(struct node));

   if(snode == NULL)
   {
      printf("\nMemory is not available");
      exit(1);
   }
   printf("\nEnter the usn,Name,programme, sem,PhoneNo of the student:");
   scanf("%s %s %s %d %ld",snode->usn, snode->name, snode->programme, &snode->sem, &snode->phone);
   snode->link=NULL;
   count++;
```

```
    return snode;
}

NODE insertfront()
{
    NODE temp;
    temp = create();
    if(start == NULL)
    {
          return temp;
    }

    temp->link = start;
    return temp;
}


NODE deletefront()
{
    NODE temp;
    if(start == NULL)
    {
       printf("\nLinked list is empty");
       return NULL;
    }

    if(start->link == NULL)
    {
          printf("\nThe Student node with usn:%s is deleted ",start->usn);
          count--;
          free(start);
          return NULL;
    }
    temp = start;
    start = start->link;
    printf("\nThe Student node with usn:%s is deleted",temp->usn);
    count--;
    free(temp);
    return start;
}

NODE insertend()
{
    NODE cur,temp;
    temp = create();

    if(start == NULL)
    {
      return temp;
    }
```

```
    cur = start;
    while(cur->link !=NULL)
    {
        cur = cur->link;
    }
    cur->link = temp;
    return start;
}

NODE deleteend()
{
    NODE cur,prev;
    if(start == NULL)
    {
      printf("\nLinked List is empty");
      return NULL;
    }

    if(start->link == NULL)
    {
      printf("\nThe student node with the usn:%s is deleted",start->usn);
      free(start);
      count--;
      return NULL;
    }

    prev = NULL;
    cur = start;
    while(cur->link!=NULL)
    {
       prev = cur;
       cur = cur->link;
    }

     printf("\nThe student node with the usn:%s is deleted",cur->usn);
     free(cur);
     prev->link = NULL;
     count--;
     return start;
}

void display()
{
    NODE cur;
    int num=1;


    if(start == NULL)
    {
```

```
        printf("\nNo Contents to display in SLL \n");
        return;
    }
    printf("\nThe contents of SLL: \n");
    cur = start;
    while(cur!=NULL)
    {
      printf("\n||%d|| USN:%s| Name:%s| programme:%s| Sem:%d| Ph:%ld|",num,cur->usn, cur->name,cur->programme, cur->sem,cur->phone);
      cur = cur->link;
      num++;
    }
    printf("\n No of student nodes is %d \n",count);
}

void stackdemo()
{
  int ch;
  while(1)
  {
   printf("\n~~~Stack Demo using SLL~~~\n");
   printf("\n1:Push operation \n2: Pop operation \n3: Display \n4:Exit \n");
   printf("\nEnter your choice for stack demo");
   scanf("%d",&ch);

   switch(ch)
   {
     case 1: start = insertfront();
             break;
     case 2: start = deletefront();
             break;
     case 3: display();
            break;
    default : return;
   }
  }
  return;
}

int main()
{
   int ch,i,n;
   while(1)
   {
      printf("\n~~~Menu~~~");
      printf("\nEnter your choice for SLL operation \n");
      printf("\n1:Create SLL of Student Nodes");
      printf("\n2:DisplayStatus");
      printf("\n3:InsertAtEnd");
      printf("\n4:DeleteAtEnd");
```

```
        printf("\n5:Stack Demo using SLL(Insertion and Deletion at Front)");
        printf("\n6:Exit \n");
        printf("\nEnter your choice:");
        scanf("%d",&ch);

        switch(ch)
        {
        case 1 : printf("\nEnter the no of students:     ");
                 scanf("%d",&n);
                 for(i=1;i<=n;i++)
                    start = insertfront();
                 break;

        case 2: display();
                break;

        case 3: start = insertend();
                break;

        case 4: start = deleteend();
                break;

        case 5: stackdemo();
                break;

        case 6: exit(0);

        default: printf("\nPlease enter the valid choice");

        }
    }
}
```

Output:

```
~~~Menu~~~
Enter your choice for SLL operation

1:Create SLL of Student Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:Stack Demo using SLL(Insertion and Deletion at Front)
6:Exit

Enter your choice:1

Enter the no of students:    2
```

```
Enter the usn,Name,programme, sem,PhoneNo of the student:1
Varun
CSE
3
9987996545

Enter the usn,Name,programme, sem,PhoneNo of the student:2
Arun
ISE
5
9656554512

~~~Menu~~~
Enter your choice for SLL operation

1:Create SLL of Student Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:Stack Demo using SLL(Insertion and Deletion at Front)
6:Exit

Enter your choice:2

The contents of SLL:

||1|| USN:2| Name:Arun| programme:ISE| Sem:5| Ph:1066619920|
||2|| USN:1| Name:Varun| programme:CSE| Sem:3| Ph:1398061953|
 No of student nodes is 2

~~~Menu~~~
Enter your choice for SLL operation

1:Create SLL of Student Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:Stack Demo using SLL(Insertion and Deletion at Front)
6:Exit

Enter your choice:4

The student node with the usn:1 is deleted
~~~Menu~~~
Enter your choice for SLL operation

1:Create SLL of Student Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
```

5:Stack Demo using SLL(Insertion and Deletion at Front)
6:Exit

Enter your choice:2

The contents of SLL:

||1|| USN:2| Name:Arun| programme:ISE| Sem:5| Ph:1066619920|
No of student nodes is 1

8. Develop a menu driven Program in C for the following operations on Doubly Linked List (DLL) of Employee Data with the fields: SSN, Name, Dept, Designation,
Sal, PhNo
a. Create a DLL of N Employees Data by using end insertion.
b. Display the status of DLL and count the number of nodes in it
c. Perform Insertion and Deletion at End of DLL
d. Perform Insertion and Deletion at Front of DLL
e. Demonstrate how this DLL can be used as Double Ended Queue.
f. Exit

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

struct node
{
        char ssn[25],name[25],dept[10],designation[25];
        int sal;
        long int phone;
        struct node *llink;
        struct node *rlink;
};
typedef struct node* NODE;

NODE first = NULL;
int count=0;


NODE create()
{
         NODE enode;
         enode = (NODE)malloc(sizeof(struct node));
         if( enode== NULL)
         {
                 printf("\nRunning out of memory");
                 exit(0);
          }
          printf("\nEnter the ssn,Name,Department,Designation,Salary,PhoneNo of the employee: \n");
```

```
        scanf("%s %s %s %s %d %ld", enode->ssn, enode->name, enode->dept, enode->designation, &enode->sal, &enode->phone);
         enode->llink=NULL;
         enode->rlink=NULL;
         count++;
         return enode;
}

NODE insertfront()
{
        NODE  temp;
        temp = create();
        if(first == NULL)
        {
              return temp;
         }
         temp->rlink = first;
         first->llink = temp;
         return temp;
}

void display()
{
         NODE cur;
         int nodeno=1;
         cur = first;
         if(cur == NULL)
                  printf("\nNo Contents to display in DLL");
         while(cur!=NULL)
        {
       printf("\nENode:%d||SSN:%s|Name:%s|Department:%s|Designation:%s|Salary:%d|Phone no:%ld", nodeno, cur->ssn, cur->name,cur->dept, cur->designation, cur->sal, cur->phone);
                 cur = cur->rlink;
                  nodeno++;
        }
         printf("\nNo of employee nodes is %d",count);
}

NODE deletefront()
{
        NODE temp;
        if(first == NULL)
        {
                 printf("\nDoubly Linked List is empty");
                 return NULL;
        }
        if(first->rlink== NULL)
       {
                 printf("\nThe employee node with the ssn:%s is deleted", first->ssn);
                free(first);
```

```
            count--;
            return NULL;
        }
        temp = first;
        first = first->rlink;
        temp->rlink = NULL;
        first->llink = NULL;
        printf("\nThe employee node with the ssn:%s is deleted",temp->ssn);
        free(temp);
        count--;
        return first;
}

NODE insertend()
{
        NODE cur, temp;
        temp = create();

        if(first == NULL)
        {
            return temp;
        }
        cur= first;
        while(cur->rlink!=NULL)
        {
            cur = cur->rlink;
        }

        cur->rlink = temp;
        temp->llink = cur;
        return first;
}

NODE deleteend()
{
        NODE prev,cur;
        if(first == NULL)
        {
            printf("\nDoubly Linked List is empty");
            return NULL;
        }

       if(first->rlink == NULL)
        {
            printf("\nThe employee node with the ssn:%s is deleted",first->ssn);
            free(first);
            count--;
            return NULL;
        }
```

```
        prev=NULL;
        cur=first;

       while(cur->rlink!=NULL)
       {
              prev=cur;
              cur = cur->rlink;
        }

        cur->llink = NULL;
        printf("\nThe employee node with the ssn:%s is deleted",cur->ssn);
        free(cur);
        prev->rlink = NULL;
        count--;
        return first;
}

void deqdemo()
{
      int ch;
      while(1)
     {
           printf("\nDemo Double Ended Queue Operation");
     printf("\n1:InsertQueueFront\n 2: DeleteQueueFront\n 3:InsertQueueRear\n 4:DeleteQueueRear\n
5:DisplayStatus\n 6: Exit \n");
           scanf("%d", &ch);

           switch(ch)
           {
                case 1: first=insertfront();
                          break;
                case 2: first=deletefront();
                          break;
                case 3: first=insertend();
                          break;
                case 4: first=deleteend();
                         break;
                case 5: display();
                          break;
                default : return;
          }
    }
}

void main()
{
   int ch,i,n;
   while(1)
   {
      printf("\n\n~~~Menu~~~");
```

```
        printf("\n1:Create DLL of Employee Nodes");
        printf("\n2:DisplayStatus");
        printf("\n3:InsertAtEnd");
        printf("\n4:DeleteAtEnd");
        printf("\n5:InsertAtFront");
        printf("\n6:DeleteAtFront");
         printf("\n7:Double Ended Queue Demo using DLL");
         printf("\n8:Exit \n");
         printf("\nPlease enter your choice: ");
        scanf("%d",&ch);

        switch(ch)
        {
         case 1 : printf("\nEnter the no of Employees:  ");
                  scanf("%d",&n);
                  for(i=1;i<=n;i++)
                  first = insertend();
                  break;

         case 2:  display();
                  break;

          case 3: first = insertend();
                 break;

          case 4: first = deleteend();
                 break;

          case 5: first = insertfront();
                 break;

          case 6: first = deletefront();
               break;

          case 7: deqdemo();
                 break;

          case 8 : exit(0);
        default: printf("\nPlease Enter the valid choice");
        }
    }
}
```

Output:

```
~~~Menu~~~
1:Create DLL of Employee Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:InsertAtFront
```

```
6:DeleteAtFront
7:Double Ended Queue Demo using DLL
8:Exit

Please enter your choice: 1

Enter the no of Employees:  2

Enter the ssn,Name,Department,Designation,Salary,PhoneNo of the employee:
11
Shamika
CSE
AP
50000
9875454545

Enter the ssn,Name,Department,Designation,Salary,PhoneNo of the employee:
12
Varun
ISE
HOD
90000
9856457845


~~~Menu~~~
1:Create DLL of Employee Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:InsertAtFront
6:DeleteAtFront
7:Double Ended Queue Demo using DLL
8:Exit

Please enter your choice: 2

ENode:1||SSN:11|Name:Shamika|Department:CSE|Designation:AP|Salary:50000|Phone no:1285519953
ENode:2||SSN:12|Name:Varun|Department:ISE|Designation:HOD|Salary:90000|Phone no:1266523253
No of employee nodes is 2

~~~Menu~~~
1:Create DLL of Employee Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:InsertAtFront
6:DeleteAtFront
7:Double Ended Queue Demo using DLL
8:Exit
```

Please enter your choice: 6

The employee node with the ssn:11 is deleted

~~~Menu~~~
1:Create DLL of Employee Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:InsertAtFront
6:DeleteAtFront
7:Double Ended Queue Demo using DLL
8:Exit

Please enter your choice: 2

ENode:1||SSN:12|Name:Varun|Department:ISE|Designation:HOD|Salary:90000|Phone no:1266523253
No of employee nodes is 1

9. Develop a Program in C for the following operationson Singly Circular Linked List (SCLL) with header nodes
a. Represent and Evaluate a Polynomial P(x,y,z) = 6x2y2z-4yz5+3x3yz+2xy5z-2xyz3
b. Find the sum of two polynomials POLY1(x,y,z) and POLY2(x,y,z) and store the result in POLYSUM(x,y,z)
Support the program with appropriate functions for each of the above operations

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#define COMPARE(x, y)      ( (x == y) ? 0 : (x > y) ? 1 : -1)

struct node
{
        int coef;
        int xexp, yexp, zexp;
        struct node *link;
};
typedef struct node *NODE;

NODE getnode()
{
        NODE x;
        x = (NODE) malloc(sizeof(struct node));
        if(x == NULL)
        {
                printf("Running out of memory \n");
                return NULL;
```
~~~

```
        }
        return x;
}

NODE attach(int coef, int xexp, int yexp, int zexp, NODE head)
{
        NODE temp, cur;
        temp = getnode();
        temp->coef = coef;
        temp->xexp = xexp;
        temp->yexp = yexp;
        temp->zexp = zexp;
        cur = head->link;
        while(cur->link != head)
        {
                cur = cur->link;
        }
        cur->link = temp;
        temp->link = head;
        return head;
}

NODE read_poly(NODE head)
{
        int i, j, coef, xexp, yexp, zexp, n;
        printf("\nEnter the no of terms in the polynomial: ");
        scanf("%d", &n);
        for(i=1; i<=n; i++)
        {
                printf("\n\tEnter the %d term: ",i);
                printf("\n\t\tCoef = ");
                scanf("%d", &coef);
                printf("\n\t\tEnter Pow(x) Pow(y) and Pow(z): ");
                scanf("%d", &xexp);
                scanf("%d", &yexp);
                scanf("%d", &zexp);
                head = attach(coef, xexp, yexp, zexp, head);
        }
         return head;
}

void display(NODE head)
{
        NODE temp;
        if(head->link == head)
        {
                 printf("\nPolynomial does not exist.");
                 return;
        }
        temp = head->link;
```

```
        while(temp != head)
        {
                printf("%dx^%dy^%dz^%d", temp->coef, temp->xexp, temp->yexp, temp->zexp);
                temp = temp->link;
                if(temp != head)
                        printf(" + ");
        }
}

int poly_evaluate(NODE head)
{
        int x, y, z, sum = 0;
        NODE poly;

        printf("\nEnter the value of x,y and z: ");
        scanf("%d %d %d", &x, &y, &z);

        poly = head->link;
        while(poly != head)
        {
                sum += poly->coef * pow(x,poly->xexp)* pow(y,poly->yexp) * pow(z,poly->zexp);
                poly = poly->link;
        }
        return sum;
}

NODE poly_sum(NODE head1, NODE head2, NODE head3)
{
   NODE a, b;
   int coef;
   a = head1->link;
   b = head2->link;

   while(a!=head1 && b!=head2)
   {
        while(1)
        {
                if(a->xexp == b->xexp && a->yexp == b->yexp && a->zexp == b->zexp)
                {
                        coef = a->coef + b->coef;
                        head3 = attach(coef, a->xexp, a->yexp, a->zexp, head3);
                        a = a->link;
                        b = b->link;
                        break;
                } //if ends here
                if(a->xexp!=0 || b->xexp!=0)
                {
                        switch(COMPARE(a->xexp, b->xexp))
                        {
```

```
                case -1 : head3 = attach(b->coef, b->xexp, b->yexp, b->zexp, head3);
                          b = b->link;
                          break;

                case 0 :  if(a->yexp > b->yexp)
                          {
                                head3 = attach(a->coef, a->xexp, a->yexp, a->zexp, head3);
                                a = a->link;
                                break;
                          }
                          else if(a->yexp < b->yexp)
                          {
                                head3 = attach(b->coef, b->xexp, b->yexp, b->zexp, head3);
                                b = b->link;
                                break;
                          }
                          else if(a->zexp > b->zexp)
                          {
                                head3 = attach(a->coef, a->xexp, a->yexp, a->zexp, head3);
                                a = a->link;
                                break;
                          }
                          else if(a->zexp < b->zexp)
                          {
                                head3 = attach(b->coef, b->xexp, b->yexp, b->zexp, head3);
                                b = b->link;
                                break;
                          }
                case 1 :  head3 = attach(a->coef,a->xexp,a->yexp,a->zexp,head3);
                          a = a->link;
                          break;
                } //switch ends here
                break;
        } //if ends here
        if(a->yexp!=0 || b->yexp!=0)
        {
                switch(COMPARE(a->yexp, b->yexp))
                {
                    case -1 :  head3 = attach(b->coef, b->xexp, b->yexp, b->zexp, head3);
                               b = b->link;
                               break;
                     case 0 :  if(a->zexp > b->zexp)
                               {
                                    head3 = attach(a->coef, a->xexp, a->yexp, a->zexp, head3);
                                    a = a->link;
                                    break;
                               }
                               else if(a->zexp < b->zexp)
                               {
                                    head3 = attach(b->coef, b->xexp, b->yexp, b->zexp, head3);
```

```
                                        b = b->link;
                                        break;
                                }
                        case 1 :    head3 = attach(a->coef, a->xexp, a->yexp, a->zexp, head3);
                                    a = a->link;
                                    break;
                        }
                        break;
                }
                if(a->zexp!=0 || b->zexp!=0)
                {
                        switch(COMPARE(a->zexp,b->zexp))
                        {
                                case -1 :  head3 = attach(b->coef,b->xexp,b->yexp,b->zexp,head3);
                                           b = b->link;
                                           break;
                                case 1 :   head3 = attach(a->coef, a->xexp, a->yexp, a->zexp, head3);
                                           a = a->link;
                                           break;
                        }
                        break;
                }
        }
    }

    while(a!= head1)
    {
          head3 = attach(a->coef,a->xexp,a->yexp,a->zexp,head3);
         a = a->link;
    }
    while(b!= head2)
    {
         head3 = attach(b->coef,b->xexp,b->yexp,b->zexp,head3);
         b = b->link;
    }
    return head3;
}


void main()
{
  NODE head, head1, head2, head3;
  int res, ch;
  head =  getnode();     /* For polynomial evalaution */
  head1 = getnode();     /* To hold POLY1 */
  head2 = getnode();     /* To hold POLY2 */
  head3 = getnode();     /* To hold POLYSUM */

  head->link=head;
  head1->link=head1;
```

```
head2->link=head2;
```

```
 head3->link= head3;

 while(1)
 {
     printf("\n~~~Menu~~~");
     printf("\n1.Represent and Evaluate a Polynomial P(x,y,z)");
     printf("\n2.Find the sum of two polynomials POLY1(x,y,z)");
     printf("\nEnter your choice:");
     scanf("%d",&ch);
     switch(ch)
     {
        case 1:         printf("\n~~~~Polynomial evaluation P(x,y,z)~~~\n");
                        head = read_poly(head);
                        printf("\nRepresentation of Polynomial for evaluation: \n");
                        display(head);
                        res = poly_evaluate(head);
                        printf("\nResult of polynomial evaluation is : %d \n", res);
                        break;

        case 2:         printf("\nEnter the POLY1(x,y,z): \n");
                        head1 = read_poly(head1);
                        printf("\nPolynomial 1 is: \n");
                        display(head1);

                        printf("\nEnter the POLY2(x,y,z): \n");
                        head2 = read_poly(head2);
                        printf("\nPolynomial 2 is: \n");
                        display(head2);

                        printf("\nPolynomial addition result: \n");
                        head3 = poly_sum(head1,head2,head3);
                        display(head3);
                        break;
           case 3:      exit(0);
     }
   }
}
```

Output:

```
~~~Menu~~~
1.Represent and Evaluate a Polynomial P(x,y,z)
2.Find the sum of two polynomials POLY1(x,y,z)
Enter your choice:1

~~~~Polynomial evaluation P(x,y,z)~~~

Enter the no of terms in the polynomial: 2

      Enter the 1 term:
           Coef =  3
```

```
        Enter Pow(x) Pow(y) and Pow(z): 1 2 3

    Enter the 2 term:
        Coef =  4

        Enter Pow(x) Pow(y) and Pow(z): 2 3 4

Representation of Polynomial for evaluation:
3x^1y^2z^3 + 4x^2y^3z^4
Enter the value of x,y and z: 2 2 2

Result of polynomial evaluation is : 2240

~~~Menu~~~
1.Represent and Evaluate a Polynomial P(x,y,z)
2.Find the sum of two polynomials POLY1(x,y,z)
Enter your choice:2

Enter the POLY1(x,y,z):

Enter the no of terms in the polynomial: 2

    Enter the 1 term:
        Coef =  7

        Enter Pow(x) Pow(y) and Pow(z): 1 2 3

    Enter the 2 term:
        Coef =  3

        Enter Pow(x) Pow(y) and Pow(z): 2 3 4

Polynomial 1 is:
7x^1y^2z^3 + 3x^2y^3z^4
Enter the POLY2(x,y,z):

Enter the no of terms in the polynomial: 2

    Enter the 1 term:
        Coef =  1

        Enter Pow(x) Pow(y) and Pow(z): 1 2 3

    Enter the 2 term:
        Coef =  2

        Enter Pow(x) Pow(y) and Pow(z): 2 3 4

Polynomial 2 is:
```

1x^1y^2z^3 + 2x^2y^3z^4
Polynomial addition result:
8x^1y^2z^3 + 5x^2y^3z^4

10. Develop a menu driven Program in C for the following operations on Binary Search Tree (BST) of Integers .
a. Create a BST of N Integers: 6, 9, 5, 2, 8, 15, 24, 14, 7, 8, 5, 2
b. Traverse the BST in Inorder, Preorder and Post Order
c. Search the BST for a given element (KEY) and report the appropriate message
d. Exit

```
#include<stdio.h>
#include<stdlib.h>
struct BST
{
        int data;
        struct BST *lchild;
        struct BST *rchild;
};
typedef struct BST * NODE;

NODE create()
{
        NODE temp;
        temp = (NODE) malloc(sizeof(struct BST));
        printf("\nEnter The value: ");
        scanf("%d", &temp->data);


        temp->lchild = NULL;
        temp->rchild = NULL;
        return temp;
}

void insert(NODE root, NODE newnode);
void inorder(NODE root);
void preorder(NODE root);
void postorder(NODE root);
void search(NODE root);

void insert(NODE root, NODE newnode)
{
   /*Note: if newnode->data ==  root->data it will be skipped. No duplicate nodes are allowed */

        if (newnode->data < root->data)
        {
                if (root->lchild == NULL)
                        root->lchild = newnode;
                else
```

```
                insert(root->lchild, newnode);
        }
        if (newnode->data > root->data)
        {
                if (root->rchild == NULL)
                        root->rchild = newnode;
                else
                        insert(root->rchild, newnode);
        }
}


void search(NODE root)
{
        int key;
        NODE cur;
        if(root == NULL)
        {
                printf("\nBST is empty.");
                return;
        }
        printf("\nEnter Element to be searched: ");
        scanf("%d", &key);
        cur = root;
        while (cur != NULL)
        {
                if (cur->data == key)
                {
                        printf("\nKey element is present in BST");
                        return;
                }
                if (key < cur->data)
                        cur = cur->lchild;
                else
                        cur = cur->rchild;
        }

        printf("\nKey element is not found in the BST");
}

void inorder(NODE root)
{
        if(root != NULL)
        {
                inorder(root->lchild);
                printf("%d ", root->data);
                inorder(root->rchild);
        }
}
```

void preorder(NODE root)

```
{
        if (root != NULL)
        {
                printf("%d ", root->data);
                preorder(root->lchild);
                preorder(root->rchild);
        }
}

void postorder(NODE root)
{
        if (root != NULL)
        {
                postorder(root->lchild);
                postorder(root->rchild);
                printf("%d ", root->data);
        }
}


void main()
{
        int ch, key, val, i, n;
        NODE root = NULL, newnode;
        while(1)
        {
                printf("\n~~~~BST MENU~~~~");
                printf("\n1.Create a BST");
                printf("\n2.BST Traversals:");
                printf("\n3. Search");
                printf("\n4.Exit");
                printf("\nEnter your choice: ");
                scanf("%d", &ch);
                switch(ch)
                {
                        case 1:         printf("\nEnter the number of elements: ");
                                        scanf("%d", &n);
                                        for(i=1;i<=n;i++)
                                        {
                                                newnode = create();
                                                if (root == NULL)
                                                        root = newnode;
                                                else
                                                        insert(root, newnode);
                                        }
                                        break;

                        case 2:          if (root == NULL)
                                                printf("\nTree Is Not Created");
                                        else
```

{

```
                                        printf("\nThe Preorder display : ");
                                        preorder(root);
                                        printf("\nThe Inorder display : ");
                                        inorder(root);
                                        printf("\nThe Postorder display : ");
                                        postorder(root);
                                }

                                break;
                        case 3:         search(root);
                                        break;

                        case 4:     exit(0);
                }
        }
}
```

Output:

```
~~~~BST MENU~~~~
1.Create a BST
2.Search
3.BST Traversals:
4.Exit
Enter your choice: 1

Enter the number of elements: 12
Enter The value: 6
Enter The value: 9
Enter The value: 5
Enter The value: 2
Enter The value: 8
Enter The value: 15
Enter The value: 24
Enter The value: 14
Enter The value: 7
Enter The value: 8
Enter The value: 5
Enter The value: 2

~~~~BST MENU~~~~
1.Create a BST
2.Search
3.BST Traversals:
4.Exit
Enter your choice: 3

The Preorder display:   6   5   2   9   8   7   15   14   24
The Inorder display:    2   5   6   7   8   9   14   15   24
```

The Postorder display: 2 5 7 8 14 24 15 9 6

~~~~BST MENU~~~~
1.Create a BST
2.Search
3.BST Traversals:
4.Exit
Enter your choice: 2

Enter Element to be searched: 66
Key element is not found in the BST

~~~~BST MENU~~~~
1.Create a BST
2.Search
3.BST Traversals:
4.Exit
Enter your choice: 2

Enter Element to be searched: 14
Key element is present in BST

11. Develop a Program in C for the following operations on Graph(G) of Cities
a. Create a Graph of N cities using Adjacency Matrix.
b. Print all the nodes reachable from a given starting node in a digraph using DFS/BFS method

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

int a[50][50], n, visited[50];
int q[20], front = -1,rear = -1;
int s[20], top = -1, count=0;

void bfs(int v)
{
        int i, cur;
        visited[v] = 1;
        q[++rear] = v;
        while(front!=rear)
        {
              cur = q[++front];
              for(i=1;i<=n;i++)
              {
                    if((a[cur][i]==1)&&(visited[i]==0))
                    {
                          q[++rear] = i;
                           visited[i] = 1;
                           printf("%d ", i);
```

```
                }
            }
        }
}

void dfs(int v)
{
        int i;
        visited[v]=1;
        s[++top] = v;
        for(i=1;i<=n;i++)
        {
             if(a[v][i] == 1&& visited[i] == 0 )
             {
                    printf("%d ", i);
                    dfs(i);
              }
        }
}

int main()
{

        int ch, start, i,j;
        printf("\nEnter the number of vertices in graph:  ");
        scanf("%d",&n);
        printf("\nEnter the adjacency matrix:\n");
        for(i=1; i<=n; i++)
        {
             for(j=1;j<=n;j++)
                  scanf("%d",&a[i][j]);
         }

   for(i=1;i<=n;i++)
         visited[i]=0;
   printf("\nEnter the starting vertex: ");
   scanf("%d",&start);

      printf("\n==>1. BFS: Print all nodes reachable from a given starting node");
      printf("\n==>2. DFS: Print all nodes reachable from a given starting node");
      printf("\n==>3:Exit");
      printf("\nEnter your choice: ");
      scanf("%d", &ch);
      switch(ch)
      {
         case 1: printf("\nNodes reachable from starting vertex %d are: ", start);
               bfs(start);
               for(i=1;i<=n;i++)
               {
                  if(visited[i]==0)
```

```
                printf("\nThe vertex that is not reachable is %d" ,i);
            }
            break;


        case 2: printf("\nNodes reachable from starting vertex %d are:\n",start);
            dfs(start);
            break;
       case 3: exit(0);
       default: printf("\nPlease enter valid choice:");
     }
}
```

Output:

```
Enter the number of vertices in graph: 5

Enter the adjacency matrix:
0 1 0 0 0
0 0 1 0 0
0 0 0 0 0
0 0 1 0 1
1 0 0 0 0

Enter the starting vertex: 1

==>1. BFS: Print all nodes reachable from a given starting node
==>2. DFS: Print all nodes reachable from a given starting node
==>3:Exit
Enter your choice: 1

Nodes reachable from starting vertex 1 are: 2 3
The vertex that is not reachable is 4
The vertex that is not reachable is 5


Enter the number of vertices in graph: 5

Enter the adjacency matrix:
0 1 0 0 0
0 0 1 0 0
0 0 0 0 0
0 0 1 0 1
1 0 0 0 0

Enter the starting vertex: 1

==>1. BFS: Print all nodes reachable from a given starting node
==>2. DFS: Print all nodes reachable from a given starting node
==>3:Exit
```

Enter your choice: 2

Nodes reachable from starting vertex 1 are:
2 3

12. Given a File of N employee records with a set K of Keys (4-digit) which uniquely determine the records in file F. Assume that file F is maintained in memory by a Hash Table (HT) of m memory locations with L as the set of memory addresses (2-digit) of locations in HT. Let the keys in K and addresses in L are Integers. Develop a Program in C that uses Hash function H: K →L as H(K)=K mod m (remainder method), and implement hashing technique to map a given key K to the address space L. Resolve the collision (if any) using linear probing.

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

int key[20],n,m;
int  *ht,index;
int count = 0;

void insert(int key)
{
        index = key % m;
        while(ht[index] != -1)
        {
                index = (index+1)%m;
        }
        ht[index] = key;
        count++;
 }

void display()
{
       int i;
       if(count == 0)
       {
                printf("\nHash Table is empty");
                return;
        }

       printf("\nHash Table contents are:\n ");
       for(i=0; i<m; i++)
              printf("\n T[%d] --> %d ", i, ht[i]);
}


void main()
{
      int i;
```

```
        printf("\nEnter the number of employee  records (N) :  ");
        scanf("%d", &n);

        printf("\nEnter the two digit memory locations (m) for hash table:  ");
        scanf("%d", &m);

        ht = (int *)malloc(m*sizeof(int));
        for(i=0; i<m; i++)
                 ht[i] = -1;

        printf("\nEnter the four digit key values (K) for N Employee Records:\n ");
        for(i=0; i<n; i++)
                 scanf("%d", &key[i]);

        for(i=0;i<n;i++)
        {
                if(count == m)
                {
                    printf("\n~~~Hash table is full. Cannot insert the record %d key~~~",i+1);
                    break;
                }
                insert(key[i]);
    }

          //Displaying Keys inserted into hash table
           display();
}
```

Output:

```
Enter the number of employee  records (N) :  5
Enter the two digit memory locations (m) for hash table: 10
Enter the four digit key values (K) for N Employee Records:
1111
1234
8545
5645
2223

Hash Table contents are:

 T[0] --> -1
 T[1] --> 1111
 T[2] --> -1
 T[3] --> 2223
 T[4] --> 1234
 T[5] --> 8545
 T[6] --> 5645
 T[7] --> -1
 T[8] --> -1
 T[9] --> -1
```

Viva Questions:

1) What is data structure?

Data structure refers to the way data is organized and manipulated. It seeks to find ways to make data access more efficient. When dealing with the data structure, we not only focus on one piece of data but the different set of data and how they can relate to one another in an organized manner.

2) Differentiate between file and structure storage structure.
The key difference between both the data structure is the memory area that is being accessed. When dealing with the structure that resides the main memory of the computer system, this is referred to as storage structure. When dealing with an auxiliary structure, we refer to it as file structures.

3) When is a binary search best applied?
A binary search is an algorithm that is best applied to search a list when the elements are already in order or sorted. The list is searched starting in the middle, such that if that middle value is not the target search key, it will check to see if it will continue the search on the lower half of the list or the higher half. The split and search will then continue in the same manner.

4) What is a linked list?
A linked list is a sequence of nodes in which each node is connected to the node following it. This forms a chain-like link for data storage.

5) How do you reference all the elements in a one-dimension array?
To reference all the elements in an one -dimension array, you need to use an indexed loop, So that, the counter runs from 0 to the array size minus one. In this manner, You can reference all the elements in sequence by using the loop counter as the array subscript.

6) In what areas do data structures are applied?
Data structures are essential in almost every aspect where data is involved. In general, algorithms that involve efficient data structure is applied in the following areas: numerical analysis, operating system, A.I., compiler design, database management, graphics, and statistical analysis, to name a few.

7) What is LIFO?
LIFO is a short form of Last In First Out. It refers how data is accessed, stored and retrieved. Using this scheme, data that was stored last should be the one to be extracted first. This also means that in order to gain access to the first data, all the other data that was stored before this first data must first be retrieved and extracted.

8) What is a queue?
A queue is a data structure that can simulate a list or stream of data. In this structure, new elements are inserted at one end, and existing elements are removed from the other end.

9) What are binary trees?
A binary tree is one type of data structure that has two nodes, a left node, and a right node. In programming, binary trees are an extension of the linked list structures.

10) Which data structures are applied when dealing with a recursive function?
Recursion, is a function that calls itself based on a terminating condition, makes use of the stack. Using LIFO, a call to a recursive function saves the return address so that it knows how to return to the calling function after the call terminates.

11) What is a stack?
A stack is a data structure in which only the top element can be accessed. As data is stored in the stack, each data is pushed downward, leaving the most recently added data on top.

12) Explain Binary Search Tree
A binary search tree stores data in such a way that they can be retrieved very efficiently. The left subtree contains nodes whose keys are less than the node's key value, while the right subtree contains nodes whose keys are greater than or equal to the node's key value. Moreover, both subtrees are also binary search trees.

13) What are multidimensional arrays?
Multidimensional arrays make use of multiple indexes to store data. It is useful when storing data that cannot be represented using single dimensional indexing, such as data representation in a board game, tables with data stored in more than one column.

14) Are linked lists considered linear or non-linear data structures?
It depends on where you intend to apply linked lists. If you based it on storage, a linked list is considered non-linear. On the other hand, if you based it on access strategies, then a linked list is considered linear.

15) How does dynamic memory allocation help in managing data?
Apart from being able to store simple structured data types, dynamic memory allocation can combine separately allocated structured blocks to form composite structures that expand and contract as needed.

16) What is FIFO?
FIFO stands for First-in, First-out, and is used to represent how data is accessed in a queue. Data has been inserted into the queue list the longest is the one that is removed first.

17) What is an ordered list?
An ordered list is a list in which each node's position in the list is determined by the value of its key component, so that the key values form an increasing sequence, as the list is traversed.

18) What is merge sort?
Merge sort, is a divide-and-conquer approach for sorting the data. In a sequence of data, adjacent ones are merged and sorted to create bigger sorted lists. These sorted lists are then merged again to form an even bigger sorted list, which continues until you have one single sorted list.

19) Differentiate NULL and VOID
Null is a value, whereas Void is a data type identifier. A variable that is given a Null value indicates an empty value. The void is used to identify pointers as having no initial size.

20) What is the primary advantage of a linked list?
A linked list is an ideal data structure because it can be modified easily. This means that editing a linked list works regardless of how many elements are in the list.

21) What is the difference between a PUSH and a POP?
Pushing and popping applies to the way data is stored and retrieved in a stack. A push denotes data being added to it, meaning data is being "pushed" into the stack. On the other hand, a pop denotes data retrieval, and in particular, refers to the topmost data being accessed.

22) What is a linear search?
A linear search refers to the way a target key is being searched in a sequential data structure. In this method, each element in the list is checked and compared against the target key. The process is repeated until found or if the end of the file has been reached.

23) How does variable declaration affect memory allocation?
The amount of memory to be allocated or reserved would depend on the data type of the variable being declared. For example, if a variable is declared to be of integer type, then 32 bits of memory storage will be reserved for that variable.

24) What is the advantage of the heap over a stack?
The heap is more flexible than the stack. That's because memory space for the heap can be dynamically allocated and de-allocated as needed. However, the memory of the heap can at times be slower when compared to that stack.

25) What is a postfix expression?
A postfix expression is an expression in which each operator follows its operands. The advantage of this form is that there is no need to group sub-expressions in parentheses or to consider operator precedence.

26) What is Data abstraction?
Data abstraction is a powerful tool for breaking down complex data problems into manageable chunks. This is applied by initially specifying the data objects involved and the operations to be performed on these data objects without being overly concerned with how the data objects will be represented and stored in memory.

27) How do you insert a new item in a binary search tree?
Assuming that the data to be inserted is a unique value (that is, not an existing entry in the tree), check first if the tree is empty. If it's empty, just insert the new item in the root node. If it's not empty, refer to the new item's key. If it's smaller than the root's key, insert it into the root's left subtree, otherwise, insert it into the root's right subtree.

28) How does a selection sort work for an array?
The selection sort is a fairly intuitive sorting algorithm, though not necessarily efficient. In this process, the smallest element is first located and switched with the element at subscript zero, thereby placing the smallest element in the first position.
The smallest element remaining in the subarray is then located next to subscripts 1 through n-1 and switched with the element at subscript 1, thereby placing the second smallest element in the second position. The steps are repeated in the same manner till the last element.

29) How do signed and unsigned numbers affect memory?
In the case of signed numbers, the first bit is used to indicate whether positive or negative, which leaves you with one bit short. With unsigned numbers, you have all bits available for that number. The effect is best seen in the number range (an unsigned 8-bit number has a range 0-255, while the 8-bit signed number has a range -128 to +127.

30) What is the minimum number of nodes that a binary tree can have?
A binary tree can have a minimum of zero nodes, which occurs when the nodes have NULL values. Furthermore, a binary tree can also have 1 or 2 nodes.

31) What are dynamic data structures?
Dynamic data structures are structures that expand and contract as a program runs. It provides a flexible means of manipulating data because it can adjust according to the size of the data.

32) In what data structures are pointers applied?
Pointers that are used in linked list have various applications in the data structure. Data structures that make use of this concept include the Stack, Queue, Linked List and Binary Tree.

33) Do all declaration statements result in a fixed reservation in memory?
Most declarations do, with the exemption of pointers. Pointer declaration does not allocate memory for data, but for the address of the pointer variable. Actual memory allocation for the data comes during run-time.

34) What are ARRAYs?
When dealing with arrays, data is stored and retrieved using an index that refers to the element number in the data sequence. This means that data can be accessed in any order. In programming, an array is declared as a variable having a number of indexed elements.

35) What is the minimum number of queues needed when implementing a priority queue?
The minimum number of queues needed in this case is two. One queue is intended for sorting priorities while the other queue is used for actual storage of data.

36) Which sorting algorithm is considered the fastest?
There are many types of sorting algorithms: quick sort, bubble sort, balloon sort, radix sort, merge sort, etc. Not one can be considered the fastest because each algorithm is designed for a particular data structure and data set. It would depend on the data set that you would want to sort.

37) Differentiate STACK from ARRAY.
Stack follows a LIFO pattern. It means that data access follows a sequence wherein the last data to be stored when the first one to be extracted. Arrays, on the other hand, does not follow a particular order and instead can be accessed by referring to the indexed element within the array.

38) Give a basic algorithm for searching a binary search tree.
1. if the tree is empty, then the target is not in the tree, end search
2. if the tree is not empty, the target is in the tree
3. check if the target is in the root item
4. if a target is not in the root item, check if a target is smaller than the root's value
5. if a target is smaller than the root's value, search the left subtree
6. else, search the right subtree

39) What is a dequeue?
A dequeue is a double-ended queue. This is a structure wherein elements can be inserted or removed from either end.

40) What is a bubble sort and how do you perform it?
A bubble sort is one sorting technique that can be applied to data structures such as an array. It works by comparing adjacent elements and exchanges their values if they are out of order. This method lets the smaller values "bubble" to the top of the list, while the larger value sinks to the bottom.

41) What are the parts of a linked list?
A linked list typically has two parts: the head and the tail. Between the head and tail lie the actual nodes. All these nodes are linked sequentially.

42) How does selection sort work?
Selection sort works by picking the smallest number from the list and placing it at the front. This process is repeated for the second position towards the end of the list. It is the simplest sort algorithm.

43) What is a graph?
A graph is one type of data structure that contains a set of ordered pairs. These ordered pairs are also referred to as edges or arcs and are used to connect nodes where data can be stored and retrieved.

44) Differentiate linear from a nonlinear data structure.
The linear data structure is a structure wherein data elements are adjacent to each other. Examples of linear data structure include arrays, linked lists, stacks, and queues. On the other hand, a non-linear data structure is a structure wherein each data element can connect to more than two adjacent data elements. Examples of nonlinear data structure include trees and graphs.

45) What is an AVL tree?
An AVL tree is a type of binary search tree that is always in a state of partially balanced. The balance is measured as a difference between the heights of the subtrees from the root. This self-balancing tree was known to be the first data structure to be designed as such.

46) What are doubly linked lists?
Doubly linked lists are a special type of linked list wherein traversal across the data elements can be done in both directions. This is made possible by having two links in every node, one that links to the next node and another one that connects to the previous node.

47) What is Huffman's algorithm?
Huffman's algorithm is used for creating extended binary trees that have minimum weighted path lengths from the given weights. It makes use of a table that contains the frequency of occurrence for each data element.

48) What is Fibonacci search?
Fibonacci search is a search algorithm that applies to a sorted array. It makes use of a divide-and-conquer approach that can significantly reduce the time needed in order to reach the target element.

49) Briefly explain recursive algorithm.
Recursive algorithm targets a problem by dividing it into smaller, manageable sub-problems. The output of one recursion after processing one sub-problem becomes the input to the next recursive process.

50) How do you search for a target key in a linked list?
To find the target key in a linked list, you have to apply sequential search. Each node is traversed and compared with the target key, and if it is different, then it follows the link to the next node. This traversal continues until either the target key is found or if the last node is reached.