KALPATARU INSTITUTE OF TECHNOLOGY, TIPTUR

INFORMATION SCIENCE AND ENGINEERING

For third Semester BE [VTU/CBCS, 2022 Scheme]

Subject Code:

B	C	S	L	3	0	5

Data Structures Lab Manual

Name: ……..….

Branch: ……..……

USN: ……..……..…

By

Dr. Shakunthala B S

Prof. Vinaykumar V N

Laboratory Outcomes:

The student should be able to:

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

Conduct of Practical Examination:

- Experiment distribution
  - For laboratories having only one part: Students are allowed to pick one experiment from the lot with equal opportunity.
  - For laboratories having PART A and PART B: Students are allowed to pick one experiment from PART A and one experiment from PART B, with equal opportunity.
- Change of experiment is allowed only once and marks allotted for procedure to be made zero of the changed part only.
- Marks Distribution (Need to change in accordance with university regulations)
  - For laboratories having only one part – Procedure + Execution + Viva-Voce: 15+70+15 = 100 Marks
  - For laboratories having PART A and PART B
    - Part A – Procedure + Execution + Viva = 6 + 28 + 6 = 40 Marks
    - Part B – Procedure + Execution + Viva = 9 + 42 + 9 = 60 Marks

Course objectives:

CO1	**Analyze various linear and non-linear data structures**
CO2	**Demonstrate the working nature of different types of data structures and their applications**
CO3	**Use appropriate searching and sorting algorithms for the give scenario.**
CO4	**Apply the appropriate data structure for solving real world problems**

Program 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>

// Define the structure for each day of the week
struct Day {
    char *name;
    int date;
    char *activity;
};

// Define an array to store 7 days of the week
struct Day week[7];

// Function to create the calendar
void create() {
    for (int i = 0; i < 7; i++) {
        week[i].name = (char *)malloc(20 * sizeof(char));
        week[i].activity = (char *)malloc(100 * sizeof(char));

        printf("Enter the name of day %d: ", i + 1);
        scanf("%s", week[i].name);

        printf("Enter the date of day %d: ", i + 1);
        scanf("%d", &week[i].date);

        printf("Enter the activity for day %d: ", i + 1);
        getchar(); // Consume the newline character
        fgets(week[i].activity, 100, stdin);
    }
}



// Function to read data from the keyboard
void read() {
    create(); // Reusing the create function for reading data
```

```
}

// Function to display the calendar
void display() {
    printf("\nDay\tDate\tActivity\n");
    for (int i = 0; i < 7; i++) {
        printf("%s\t%d\t%s", week[i].name, week[i].date, week[i].activity);
    }
}

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

    printf("\nDisplaying the calendar:\n");
    display();

    return 0;
}
```

Output:

Creating a calendar:

Enter the name of day 1: Monday

Enter the date of day 1: 30

Enter the activity for day 1: 3rd Sem starts

Enter the name of day 2: Tuesday

Enter the date of day 2: 31

Enter the activity for day 2: Interaction with juniors

Enter the name of day 3: Wednesday

Enter the date of day 3: 1

Enter the activity for day 3: Kannada Rajyotsava

Enter the name of day 4: Thursday

Enter the date of day 4: 2

Enter the activity for day 4: Technical Talk from Industry Experts

Enter the name of day 5: Friday

Enter the date of day 5: 3

Enter the activity for day 5: Cultural Activities

Enter the name of day 6: Saturday

Enter the date of day 6: 4

Enter the activity for day 6: Hands on session (Web Designing)

Enter the name of day 7: Saturday

Enter the date of day 7: 5

Enter the activity for day 7: Refreshment

Displaying the calendar:

Day	Date	Activity
Monday	30	3rd Sem starts
Tuesday	31	Interaction with juniors
Wednesday	1	Kannada Rajyotsava
Thursday	2	Technical Talk from Industry Experts
Friday	3	Cultural Activities
Saturday	4	Hands on session (Web Designing)
Sunday	5	Refreshment

Program 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: Welcome to BGSgroups

Enter the pat string: groups

Enter the replace string: CET

Resultant string is Welcome to BGSCET

Program 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: 25
~~~~~~

~~~~~~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: 26

~~~~~~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: 27

~~~~~~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: 28
~~~~~~

~~~~~~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: 4

Stack elements are:

| 28 |

| 27 |

| 26 |

| 25 |

~~~~~~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: 2

Element popped is: 28

~~~~~~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: 4

Stack elements are:

| 27 |

| 26 |

| 25 |

~~~~~~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:

| 27 |

| 26 |

| 25 |

Reverse of stack content are:

| 25 |

| 26 |

| 27 |

It is not a palindrome number
~~~~~~

~~~~~~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: 5

Output 2:

~~~~~~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: 1

~~~~~~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: 2

~~~~~~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: 1

~~~~~~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: 4

Stack elements are:

| 1 |

| 2 |

| 1 |

~~~~~~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:

| 1 |

| 2 |

| 1 |

Reverse of stack content are:

| 1 |

| 2 |

| 1 |

It is palindrome number

~~~~~~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: 5
~~~~~~

Program 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 <ctype.h>
#include <stdio.h>

#define SIZE 50 /* Size of Stack */
char s[SIZE]; /* Global declarations */
int top = -1;

push(char elem) /* Function for PUSH operation */
{
        s[++top] = elem;
}
char pop() /* Function for POP operation */
{
        return (s[top--]);
}

int pr(char elem) /* Function for precedence */
{
        switch (elem)
        {
                case '#': return 0; case '(': return 1; case '+':
                case '-': return 2; case '*':
                case '/':
                case '%': return 3; case '^':
                return 4;
        }
}
void main() /* Main Program */
{
```

```
        char infx[50], pofx[50], ch, elem; int i = 0, k = 0;
        printf("\n\n enter the Infix Expression : ");
        gets(infx);
        push('#');
        while ((ch = infx[i++]) != '\0')
        {
                if (ch == '(')
                push(ch);
                else if (isalnum(ch))
                        pofx[k++] = ch;
                else if (ch == ')')
                {
                        while (s[top] != '(') pofx[k++] = pop();
                        elem = pop(); /* Remove ( */
                }
                else    /* Operator */
                {
                        while (pr(s[top]) >= pr(ch)) pofx[k++] = pop();
                        push(ch);
                }
        }
while (s[top] != '#') /* Pop from stack till empty */
pofx[k++] = pop();
pofx[k] = '\0'; /* Make pofx as valid string */
printf("\n\n Given Infix Expn is: %s\n The Postfix Expn is:%s\n", infx, pofx);
}
```

Output 1:

enter the Infix Expression : a+b-c*d

Given Infix Expn is: a+b-c*d

The Postfix Expn is: ab+cd*-

Output 2:

enter the Infix Expression : a+(b-c)*d/e^f

Given Infix Expn is: a+(b-c)*d/e^f

The Postfix Expn is:abc-d*ef^

Program 5:

Develop a Program in C for the following Stack Applications a. Evaluation of Suffix expression with single digit operands and operators: +, -, *, /, %, ^

```
#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:

651-4*23$/+

Result = 8

Output 2:

Enter a valid postfix expression:

861-*2/32^%

Result = 2

b. Solving Tower of Hanoi problem with n disks

```
#include<stdio.h>
#include<conio.h> int count=0,n;
int tower(int n,char s,char t,char d)
{
        if(n==1)
        {
                printf("\n Move disc 1 from %c to %c",s,d); count++;
                return 1;
        }
        tower(n-1,s,d,t);
        printf("\n Move disc %d from %c to %c",n,s,d); count++;
        tower(n-1,t,s,d);
}

int main( )
{
printf("\n Enter the no. of discs:");
scanf("%d",&n);
tower(n,'A','B','C');
printf("\n The no. of disc moves is:%d",count);
getch( );
}
```

Output 1:

Enter the no. of discs:2

Move disc 1 from A to B

Move disc 2 from A to C

Move disc 1 from B to C

The no. of disc moves is:3

Output 2:

Enter the no. of discs:4

Move disc 1 from A to B

Move disc 2 from A to C

Move disc 1 from B to C

Move disc 3 from A to B

Move disc 1 from C to A

Move disc 2 from C to B

Move disc 1 from A to B

Move disc 4 from A to C

Move disc 1 from B to C

Move disc 2 from B to A

Move disc 1 from C to A

Move disc 3 from B to C

Move disc 1 from A to B

Move disc 2 from A to C

Move disc 1 from B to C

The no. of disc moves is: 15

Program 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 onto 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.

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define SIZE 5
int q[SIZE], i, r=-1, f=0, option, count=0, j;
int main( )
{
        for(;;)
        {
        printf("\n 1.Insert 2.Delete\n 3.Display 4.Exit");
        printf("\nEnter your option:");
        scanf("%d",&option);
        switch(option)
        {
                case 1: //Inserting items to Queue
                        if(count==SIZE)
                        printf("\n Q is Full\n");
                        else
                        {
                                r=(r+1)%SIZE;
                                printf("\nEnter the item:");
                                scanf("%d",&q[r]);
                                count++;
                        }
                        break;
                case 2: //Deleting items from Queue
                if(count==0)
```

```
                printf("\nQ is empty\n");
                else
                {
                        printf("\nDeleted item is: %d",q[f]);
                        count--;
                        f=(f+1)%SIZE;
                }
                break;
                case 3: //Displaying items from Queue
                if(count==0)
                printf("\nQ is Empty\n");
                else
                {
                        i=f;
                        for(j=0;j<count;j++)
                        {
                                printf(" %d",q[i]);
                                 i=(i+1)%SIZE;
                        }
                }
                break;
                default:  exit(0);
                }
        }
}
```

Output:

1.Insert 2.Delete

3.Display 4.Exit

Enter your option:1

Enter the item:10

1.Insert 2.Delete

3.Display 4.Exit

Enter your option:1

Enter the item:20

1.Insert 2.Delete

3.Display 4.Exit

Enter your option:1

Enter the item:30

1.Insert 2.Delete

3.Display 4.Exit

Enter your option:1

Enter the item:40

1.Insert 2.Delete

3.Display 4.Exit

Enter your option:1

Enter the item:50

1.Insert 2.Delete

3.Display 4.Exit

Enter your option:1

Q is Full

1.Insert 2.Delete

3.Display 4.Exit

Enter your option:3

10 20 30 40 50

1.Insert 2.Delete

3.Display 4.Exit

Enter your option:2

Deleted item is: 10

1.Insert 2.Delete

3.Display 4.Exit

Enter your option:2

Deleted item is: 20

1.Insert 2.Delete

3.Display 4.Exit

Enter your option:3

30 40 50

1.Insert 2.Delete

3.Display 4.Exit

Enter your option:1

Enter the item:60

1.Insert 2.Delete

3.Display 4.Exit

Enter your option:2

Deleted item is: 30

1.Insert 2.Delete

3.Display 4.Exit

Enter your option:2

Deleted item is: 40

1.Insert 2.Delete

3.Display 4.Exit

Enter your option:2

Deleted item is: 50

1.Insert 2.Delete

3.Display 4.Exit

Enter your option:2

Deleted item is: 60

1.Insert 2.Delete

3.Display 4.Exit

Enter your option:2

Q is empty

Program 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>
#include<string.h>
 struct student
{
        char usn[11];
        char name[25];
        int sem;
        char branch[5];
        unsigned long long phno;
};
typedef struct student STUD;

struct node
{
        char usn[11];
        char name[25];
        int sem;
        char branch[5];
        unsigned long long phno;
        struct node *next;
};
typedef struct node NODE;
NODE *first;
NODE* copyNode(STUD s)
{
        NODE * temp;
```

```
        temp= (NODE *)malloc(sizeof(NODE));
        if(temp==NULL)
        {
        printf("Memory cannot be allocated\n");
        }
        else
        {
        strcpy(temp->usn,s.usn);
        strcpy(temp->name, s.name);
        strcpy(temp->branch, s.branch);
        temp->sem=s.sem;
        temp->phno=s.phno;
        temp->next=NULL;
        return temp;
        }
}
void addrear(STUD s)
{
   NODE *temp,*cur;
   temp=copyNode(s) ;
   if(first==NULL)
   {
        temp=first;
        return;
   }
        cur=first;
while(cur->next != NULL)
{
        cur=cur->next;
}
cur->next =temp;
return ;
}
void addfront(STUD s)
```

```
{
        NODE *temp;
        temp=copyNode(s); //ssn,name, dept, design,salary, pno);
        if (first== NULL)
        {
                first=temp;
        }
        else
        {
                temp->next=first;
                first=temp;
        }
        return ;
}

void display(NODE *temp)
{
        printf("%s \t", temp->usn);
        printf("%s \t", temp->name);
        printf("%s \t", temp->branch);
        printf("%d \t",temp->sem);
        printf("%llu \n", temp->phno);
}
void deletefront()
{
        NODE *temp;
        int num;
        temp=first;
        if(first==NULL)
        {
                printf("List is Empty");
                return;
        }
        if(first->next==NULL)
```

```
        first=NULL;
        else
        {
                first=first->next;
        }
        printf("Deleted Node is:\n");
display(temp);
free(temp);
return;
}
void deleterear()
{
        NODE *cur,*prev;
        cur=first;
        prev=NULL;
        if(first==NULL)
        {
                printf("List is Empty");
                return;
        }
        if(first->next==NULL)
        {
                display(cur);
                first=NULL;
                free(cur);
                return;
        }
        while(cur->next!=NULL)
        {
                prev=cur;
                cur=cur->next;
        }
prev->next=NULL;
printf("Deleted Node is:\n");
```

```
display(cur);
free(cur);
return;
}
void  displayList()
{
        NODE *r;
        r=first;
        printf("USN\tName\tBrh\tSem\tPhone\n");
        if(r==NULL)
                return;
        while(r!=NULL)
        {
                display(r);
                r=r->next;
        }
        printf("\n");
}
STUD input()
{
STUD s;
printf("Enter USN: ");
scanf("%s",s.usn);
printf("Enter Name     : ");
scanf("%s",s.name);
printf("Enter Branch: ");
scanf("%s",s.branch);
printf("Enter Sem:");
scanf("%d",&s.sem);
printf("Enter Phone no : ");
scanf("%llu",&s.phno);
return s;
}
int count()
```

```
{
NODE *n;
int c=0;
n=first;
while(n!=NULL)
{
        n=n->next;
        c++;
}
return c;
}
int  main()
{
STUD s;
int i,ch, n;
first=NULL;
while(1)
{
        printf("\nList Operations\n");
        printf("===============\n");
        printf("1.Create List of n students by using front Insert\n");
        printf("2.Display the status and count the nodes\n");
        printf("3.Perform Insertion and Deletion at End of SLL\n");
        printf("4.Perform Insertion and Deletion at Front of SLL\n");
        printf("5.Exit\n"); printf("Enter your choice : ");
        scanf("%d",&i);
        switch(i)
        {
        case 1 : printf("Enter the number of students\n");
                scanf("%d",&n);
                for(i=1;i<=n;i++)
                {
                        printf("\nEnter the details of student %d\n",i);
                        s=input();
```

```
                addfront(s);
            }
            break;
        case 2 :  if(first==NULL)
            {
                printf("List is Empty\n");
            }
            else
            printf(" Node Count=%d\t & Elements in the list are : \n", count());
            displayList();
            }
            break;
        case 3 : printf(" 1. Insert at End and 2 Delete From End=");
            scanf("%d",&ch);
            if(ch==1)
            {
                 s=input(); addrear(s);
            }
            else if(ch==2)
            deleterear();
            else
            printf(" Sorry wrong operation\n"); break;
        case 4 : printf(" 1. Insert at Front and 2.Delete From Front=");
            scanf("%d",&ch);
            if(ch==1)
            {
                 s=input(); addfront(s);
            }
            else if(ch==2)
            deletefront();
            else
            printf(" Sorry wrong operation\n");
        case 5 :exit (0);
        default : printf("Invalid option\n");
        }
}
return 0;
}
```

Output

```
List Operations
===============
1.Create List of n students by using front Insert
2.Display the status and count the nodes
3.Perform Insertion and Deletion at End of SLL
4.Perform Insertion and Deletion at Front of SLL
5.Exit
Enter your choice : 1
Enter the number of students
2

Enter the details of student 1
Enter USN: 1mp22is001
Enter Name      : aishwarya
Enter Branch: ise
Enter Sem:3
Enter Phone no : 8765432192

Enter the details of student 2
Enter USN: 1mp22cs002
Enter Name      : abhishek
Enter Branch: cse
Enter Sem:3
Enter Phone no : 9876543213

List Operations
===============
1.Create List of n students by using front Insert
2.Display the status and count the nodes
3.Perform Insertion and Deletion at End of SLL
4.Perform Insertion and Deletion at Front of SLL
5.Exit
Enter your choice : 3
 1. Insert at End and 2 Delete From End=1
Enter USN: 1mp22ec003
Enter Name      : asha
Enter Branch: ec
```

Enter Sem:3
Enter Phone no : 8976543218

List Operations
==========
1.Create List of n students by using front Insert
2.Display the status and count the nodes
3.Perform Insertion and Deletion at End of SLL
4.Perform Insertion and Deletion at Front of SLL
5.Exit
Enter your choice : 2
Node Count=3 & Elements in the list are :

USN	Name	Brh	Sem	Phone
1mp22cs002	abhishek	cse	3	9876543213
1mp22is001	aishwarya	ise	3	8765432192
1mp22ec003	asha	ec	3	8976543218

List Operations
==========
1.Create List of n students by using front Insert
2.Display the status and count the nodes
3.Perform Insertion and Deletion at End of SLL
4.Perform Insertion and Deletion at Front of SLL
5.Exit
Enter your choice : 2
Node Count=3 & Elements in the list are :

USN	Name	Brh	Sem	Phone
1mp22cs002	abhishek	cse	3	9876543213
1mp22is001	aishwarya	ise	3	8765432192
1mp22ec003	asha	ec	3	8976543218

List Operations
==========
1.Create List of n students by using front Insert
2.Display the status and count the nodes
3.Perform Insertion and Deletion at End of SLL
4.Perform Insertion and Deletion at Front of SLL

5.Exit

Enter your choice : 4

1. Insert at Front and 2.Delete From Front=2

Deleted Node is:

1mp22cs002 abhishek cse 3 9876543213

Program 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>
#include<string.h> struct employee
{
char ssn[11];
char name[21];
char dept[15];
char design[15];
int  salary;
unsigned long long  phno;
};
typedef struct employee EMP; struct node
{
char ssn[11];
char name[21];
char dept[15];
char design[15];
int  salary;
unsigned long long phno;
struct node *next;
struct node *prev;
};
typedef struct node NODE;
NODE *first;
NODE* copyNode(EMP e)
{
NODE * temp;
temp= (NODE *)malloc(sizeof(NODE));
if(temp==NULL)
```

```
{
printf("Memory cannot be allocated\n");
}
else
{
strcpy(temp->ssn,e.ssn);
strcpy(temp->name, e.name);
strcpy(temp->dept, e.dept);
strcpy(temp->design, e.design);
temp->salary=e.salary;
temp->phno=e.phno;
temp->next=NULL;
temp->prev=NULL;
return temp;
}
}
void addrear(EMP e)
{
NODE *temp,*cur, *prev;
temp=copyNode(e) ;
if(first==NULL)
{
        first=temp;
        return;
}
cur=first;
prev=NULL;
while(cur->next != NULL)
{
        prev=cur;
        cur=cur->next;
}
cur->next =temp;
temp->prev=prev;
return ;
}
void addfront(EMP e)
{
NODE *temp;
```

```
temp=copyNode(e);
if (first== NULL)
{
        first=temp;
}
else
{
        temp->next=first;
        first->prev=temp;
        first=temp;
}
return ;
}
void display(NODE *r)
{
printf("%s\t", r->ssn);
printf("%s\t", r->name);
printf("%s\t", r->dept);
printf("%s\t",r->design);
printf("%d\t",r->salary);
printf("%llu\n", r->phno);
}
void deletefront()
{
NODE *temp;
int num;
temp=first;
if(first==NULL)
{
        printf(" List is Empty \n");
        return ;
}
if(first->next==NULL)
        first=NULL;
else
{
        first=first->next;
        first->prev=NULL;
}
```

```
printf("SSN\tName\tDept\tDesignation\tSalary\tPhone\n");
display(temp);
free(temp);
return ;
}
void deleterear()
{
NODE *cur, *prev;
cur=first;
prev=NULL;
if(first==NULL)
{
        printf(" List is Empty \n");
        return ;
}
if(first->next==NULL)
{
        first=NULL;
}
else
{
while(cur->next!=NULL)
{
        prev=cur;
        cur=cur->next;
}
prev->next=NULL;
}
printf("SSN\tName\tDept\tDesignation\tSalary\tPhone\n"); display(cur);
free(cur);
 return;
}
void displayList()
{
NODE *r;
r=first;
if(r==NULL)
{
        return;
```

```
}
printf("SSN\t Name\t Dept\t Designation\t salary\tPhone\n");
while(r!=NULL)
{
        display(r);
        r=r->next;
}
printf("\n");
}
int count()
{
NODE *n;
int c=0;
n=first;
while(n!=NULL)
{
        n=n->next; c++;
}
return c;
}
EMP input()
{
EMP e;
printf("Enter SSN: ");
scanf("%s",e.ssn);
printf("Enter Name: ");
scanf("%s",e.name);
printf("Enter dept: ");
scanf("%s",e.dept);
printf("Enter Designation :");
scanf("%s",e.design);
printf("Enter Salary:");
scanf("%d",&e.salary);
printf("Enter Phone no : ");
scanf("%llu",&e.phno);
return e;
}
int  main()
{
```

```
EMP e;
int i, ch, n;
first=NULL;
while(1)
{
printf("\nList Operations\n");
printf("===============\n");
printf("1.Create a DLL of N Employees Data by using end insertion\n");
printf("2.Display the status of DLL and count the number of nodes in it\n");
printf("3.Perform Insertion and Deletion at End of DLL\n");
printf("4.Perform Insertion and Deletion at Front of DLL\n");
printf("5.Demonstration of this DLL as Double Ended Queue\n");
printf("6.Exit\n");
printf("Enter your choice : ");
scanf("%d",&i);
switch(i)
{
case 1 :
printf("Enter the number of Employees\n");
scanf("%d",&n);
for(i=1;i<=n;i++)
{
        printf("\nEnter the details of Employee %d\n",i);
        e=input();
        addrear(e);
}
break;

case 2 :
if(first==NULL)
{
        printf("List is Empty\n");
}
else
{
        printf(" Node Count=%d\t & Elements in the list are : \n", count());
        displayList();
}
break;
```

```
        case 3 :
        printf(" 1. Insert at End and 2 Delete From End=");
        scanf("%d",&ch);
        if(ch==1)
        {
                e=input();
                addrear(e);
        }
        else if(ch==2)
                deleterear();
        else
                printf(" Sorry wrong operation\n");
                break;
        case 4 :
        printf(" 1. Insert at Front and 2 Delete From Front=");
        scanf("%d",&ch);
         if(ch==1)
        {
                e=input();
                addfront(e);
        }
        else if(ch==2)
                deletefront();
        else
        printf(" Sorry wrong operation\n");
        break;
        case 5:
        printf("This DLL can be used as Double Ended Queue by inserting and deleting
        from both ends \n");
        break;
        case 6  :
        return 0;
        default : printf("Invalid option\n");
        }
        }
return 0;
}
```

Output:

List Operations
==========
1.Create a DLL of N Employees Data by using end insertion
2.Display the status of DLL and count the number of nodes in it
3.Perform Insertion and Deletion at End of DLL
4.Perform Insertion and Deletion at Front of DLL
5.Demonstration of this DLL as Double Ended Queue
6.Exit
Enter your choice : 1
Enter the number of Employees 2
Enter the details of Employee 1
Enter SSN: 111
Enter Name: Ram
Enter dept: Sales
Enter Designation :Manager
Enter Salary:50000
Enter Phone no : 9141000000

Enter the details of Employee 2
Enter SSN: 222
Enter Name: Sham
Enter dept: Design
Enter Designation :Manager
Enter Salary:60000
Enter Phone no : 9876543210

List Operations
==========
1.Create a DLL of N Employees Data by using end insertion
2.Display the status of DLL and count the number of nodes in it
3.Perform Insertion and Deletion at End of DLL
4.Perform Insertion and Deletion at Front of DLL
5.Demonstration of this DLL as Double Ended Queue
6.Exit
Enter your choice : 2
Node Count=2 & Elements in the list are :

SSN	Name	Dept	Designation	salary	Phone
111	Ram	Sales	Manager	50000	9141000000
222	Sham	Design	Manager	60000	9876543210

List Operations

==========

1.Create a DLL of N Employees Data by using end insertion

2.Display the status of DLL and count the number of nodes in it

3.Perform Insertion and Deletion at End of DLL

4.Perform Insertion and Deletion at Front of DLL

5.Demonstration of this DLL as Double Ended Queue

6.Exit

Enter your choice : 3

1. Insert at End and 2 Delete From End=1

Enter SSN: 333

Enter Name: Ravi

Enter dept: Finance

Enter Designation :Manager

Enter Salary:65000

Enter Phone no : 9876500000

List Operations

==========

1.Create a DLL of N Employees Data by using end insertion

2.Display the status of DLL and count the number of nodes in it

3.Perform Insertion and Deletion at End of DLL

4.Perform Insertion and Deletion at Front of DLL

5.Demonstration of this DLL as Double Ended Queue

6.Exit

Enter your choice : 2

Node Count=3 & Elements in the list are :

SSN	Name	Dept	Designation	salary	Phone
111	Ram	Sales	Manager	50000	9141000000
222	Sham	Design	Manager	60000	9876543210
333	Ravi	Finance	Manager	65000	9876500000

List Operations

==========

1.Create a DLL of N Employees Data by using end insertion
2.Display the status of DLL and count the number of nodes in it
3.Perform Insertion and Deletion at End of DLL
4.Perform Insertion and Deletion at Front of DLL
5.Demonstration of this DLL as Double Ended Queue
6.Exit
Enter your choice : 4
1. Insert at Front and 2 Delete From Front=2
SSN Name Dept Designation Salary Phone
111 Ram Sales Manager 50000 9141000000

List Operations
==========
1.Create a DLL of N Employees Data by using end insertion
2.Display the status of DLL and count the number of nodes in it
3.Perform Insertion and Deletion at End of DLL
4.Perform Insertion and Deletion at Front of DLL
5.Demonstration of this DLL as Double Ended Queue
6.Exit
Enter your choice : 2
Node Count=2 & Elements in the list are :
SSN Name Dept Designation salary Phone
222 Sham Design Manager 60000 9876543210
333 Ravi Finance Manager 65000 9876500000

List Operations
==========
1.Create a DLL of N Employees Data by using end insertion
2.Display the status of DLL and count the number of nodes in it
3.Perform Insertion and Deletion at End of DLL
4.Perform Insertion and Deletion at Front of DLL
5.Demonstration of this DLL as Double Ended Queue
6.Exit
Enter your choice : 5
This DLL can be used as Double Ended Queue by inserting and deleting from both ends

List Operations
==========
1.Create a DLL of N Employees Data by using end insertion

2.Display the status of DLL and count the number of nodes in it
3.Perform Insertion and Deletion at End of DLL
4.Perform Insertion and Deletion at Front of DLL
5.Demonstration of this DLL as Double Ended Queue
6.Exit
Enter your choice : 6

Program 9:

Develop a Program in C for the following operations on 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.

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

int coef,px,py,pz, x,y,z,i; int val;

struct node
{
int coef,px, py,pz; struct node *next;
};
typedef struct node NODE; NODE *first;

void insert(int coef,int px, int py, int pz)
{
NODE *temp,*cur;
temp= (NODE *)malloc(sizeof(NODE));
temp->coef=coef;
temp->px=px;
temp->py=py;
temp->pz=pz;
if(first==NULL)
{
        temp->next=temp;
        first=temp;
        return;
}
if(first->next==first)
{
         first->next=temp;
        temp->next=first;
```

```
}
cur=first;
while(cur->next!=first)
{
        cur=cur->next;
}
cur->next=temp;
temp->next=first;
return;
}

void display()
{
NODE *cur;
if(first==NULL)
{
        printf("List is empty\n");
        return;
}
cur=first;
while(cur->next!=first)
{
 printf("%d ",cur->coef);
printf(" x^%d",cur->px);
printf(" y^%d",cur->py);
printf(" z^%d + ",cur->pz);
cur=cur->next;
}
printf("%d ",cur->coef);
printf(" x^%d",cur->px);
printf(" y^%d",cur->py);
printf(" z^%d\n",cur->pz);
return;
}
int  evaluate(int x, int y, int z)
{
NODE *cur; int v,s=0, v1,v2,v3;
if(first==NULL)
{
```

```
        printf("List is empty\n");return 0;
}
cur=first;
while(cur->next!=first)
{
        v=cur->coef*pow(x, cur->px)*pow(y, cur->py)*pow(z,cur->pz);
        s=s+v;
        cur=cur->next;
}
v=cur->coef*pow(x, cur->px)*pow(y, cur->py)*pow(z,cur->pz);
 s=s+v;
return s;
}

int main()
{
int coef,px,py,pz, x,y,z,i;
int val;
first=NULL;
while(1)
{
printf("1. Insert polynomial at end\n");
printf("2. Display\n");
printf("3. Evaluate\n");
printf("4. Exit\n");
printf("Enter Choice= \t");
scanf("%d",&i);
switch(i)
{
case 1 : printf("Enter Coefficient= \t");
        scanf("%d",&coef);
        printf("Enter powers of x y z values= \t");
        scanf("%d%d%d",&px, &py,&pz); insert(coef,px,py,pz);
break;
case 2 :
        display();
break;
case 3 : printf("\n Enter x y & z values for evaluation: \t");
        scanf("%d%d%d",&x,&y,&z);
```

```
                val=evaluate(x,y,z);
                printf("\nValue=%d\n",val);
                break;
        case 4 :  return 0;
        default : printf(" Wrong choice. Enter 1,2 3\n");
        break;
        }
}
}
```

Output:

```
1. Insert polynomial at end
2. Display
3. Evaluate
4. Exit
Enter Choice=  1
Enter Coefficient=      4
Enter powers of x y z values=   3 3 2
1. Insert polynomial at end
2. Display
3. Evaluate
4. Exit
Enter Choice=  1
Enter Coefficient=      5
Enter powers of x y z values=   3 2 2
1. Insert polynomial at end
2. Display
3. Evaluate
4. Exit
Enter Choice=  1
Enter Coefficient=      3
Enter powers of x y z values=   2 2 2
1. Insert polynomial at end
2. Display
3. Evaluate
4. Exit
Enter Choice=  1
Enter Coefficient=      6
Enter powers of x y z values=   1 1 1
```

```
1. Insert polynomial at end
2. Display
3. Evaluate
4. Exit
Enter Choice=  2
4 x^3 y^3 z^2 + 5  x^3 y^2 z^2 + 3  x^2 y^2 z^2 + 6  x^1 y^1 z^1
1. Insert polynomial at end
2. Display
3. Evaluate
4. Exit
Enter Choice=  3

 Enter x y & z values for evaluation: 3 2 1

Value=1548
1. Insert polynomial at end
2. Display
3. Evaluate
4. Exit
Enter Choice=  4
```

Program 9B

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

struct node      //Defining Polynomial fields
{
        int coef, px, py, pz,flag;
        struct node *link;
};

typedef struct node * NODE;

NODE create_list(NODE head)  //For creating poly1 & poly2
{
int  i,n,cf,px,py,pz;
printf("Enter the number of terms : ");
 scanf("%d",&n);
```

```
for(i=1;i<=n;i++)
{
        printf("Enter the Co-ef, px, py, pz : ");
        scanf("%d %d %d %d",&cf,&px,&py,&pz);
        insert(head,cf,px,py,pz);
}
return head;
}/*End of create_list()*/

insert(NODE head,int cof,int x,int y, int z) //inserting term to poly
{
NODE cur,tmp;
tmp= (NODE)malloc(sizeof(struct node)); //Allocates memory int cf,px,py,pz;
cur=head->link;
tmp->coef=cof;
tmp->px=x;
tmp->py=y;
tmp->pz=z;
tmp->flag=0;
while(cur->link!=head) //Identifying last node
        cur=cur->link;
cur->link=tmp;
tmp->link=head;
}

NODE add_poly(NODE h1,NODE h2,NODE h3)
{
NODE cur1,cur2,scf; cur1=h1->link; cur2=h2->link;
while(cur1 != h1)  //Till end of poly1
{
if(cur2 == h2)
cur2=h2->link;
while(cur2 != h2)  //Till end of poly2
{
if(cur1->px == cur2->px && cur1->py == cur2->py && cur1->pz == cur2->pz)
{       //Add & insert if co-ef's of both poly is equal
scf = cur1->coef + cur2->coef;
insert(h3,scf,cur1->px,cur1->py,cur1->pz);
cur2->flag=1;
```

```
cur2=h2->link; break;
}
cur2=cur2->link;
}
if(cur1 == h1)
break;
if(cur2 == h2) //If co-ef of poly1 is not matched, insert it to poly3
insert(h3,cur1->coef,cur1->px,cur1->py,cur1->pz);
cur1=cur1->link;
}

cur2=h2->link;
while(cur2 != h2)  //remaining poly2 nodes inserted to poly3
{
if(cur2->flag==0)
insert(h3,cur2->coef,cur2->px,cur2->py,cur2->pz);
cur2=cur2->link;
}
return h3;
}

void display(NODE head)
{
NODE cur;
if(head->link==head) //if poly is empty
{
        printf("List is empty\n"); return;
}
cur=head->link;
while(cur != head)  //display all terms till end
{
if(cur->coef > 0)
        printf(" +%dx^%dy^%dz^%d ",cur->coef,cur->px,cur->py,cur->pz);
else if (cur->coef < 0)
        printf(" %dx^%dy^%dz^%d ",cur->coef,cur->px,cur->py,cur->pz);
        cur=cur->link;
}
printf("\n");
}/*End of display() */
```

```
void main()
{
int choice,data,item,pos; NODE head1,head2,head3;

head1=(NODE)malloc(sizeof(struct node));
head1->link=head1;                    //poly1

head2=(NODE)malloc(sizeof(struct node));
head2->link=head2;                    //poly2

head3=(NODE)malloc(sizeof(struct node));
head3->link=head3;                    //poly3

printf("\n1.Create Polynomial 1\n");
head1=create_list(head1);

printf("\n2.Create Polynomial 2\n");
head2=create_list(head2);
printf("\nPolynomial 1 is :");
display(head1);

printf("\nPolynomial 2 is :");
display(head2);

head3=add_poly(head1,head2,head3); //Add both polynomials

printf("\nAddition of two Polynomial is :");
 display(head3);
}
```

Output:

1.Create Polynomial 1

Enter the number of terms : 4

Enter the Co-ef, px, py, pz : 5 3 3 3

Enter the Co-ef, px, py, pz : 6 3 3 2

Enter the Co-ef, px, py, pz : 8 2 2 2

Enter the Co-ef, px, py, pz : 9 1 0 1

2.Create Polynomial 2

Enter the number of terms : 4

Enter the Co-ef, px, py, pz : 6 3 3 3

Enter the Co-ef, px, py, pz : 5 2 2 2

Enter the Co-ef, px, py, pz : 8 2 1 2

Enter the Co-ef, px, py, pz : 9 1 1 0

Polynomial 1 is : +5x^3y^3z^3 +6x^3y^3z^2 +8x^2y^2z^2 +9x^1y^0z^1

Polynomial 2 is : +6x^3y^3z^3 +5x^2y^2z^2 +8x^2y^1z^2 +9x^1y^1z^0

Addition of two Polynomial is :

+6x^3y^3z^2 +9x^1y^0z^1 +6x^3y^3z^3 +5x^2y^2z^2 +8x^2y^1z^2 +9x^1y^1z^0

Program 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<conio.h>
#include<stdlib.h>
int choice,data,key;
struct node
{
int info;
struct node *lchild,*rchild;
};
typedef struct node *NODE;
int main()
{
NODE root=NULL;
NODE CREATE(NODE,int);
void INORDER(NODE),POSTORDER(NODE),PREORDER(NODE);
NODE SEARCH_NODE(NODE,int);

while(1)
{
printf("\n1:CREATE\n2:TREE TRAVERSAL\n3.SEARCH\n4.EXIT");
printf("\nEnter your choice\n");
scanf("%d",&choice);
switch(choice)
{
case 1: printf("\nEnter data to be inserted\n");
scanf("%d",&data);
root=CREATE(root,data);
break;
case 2: if(root==NULL)
printf("\nEMPTY TREE\n");
else
```

```
{
printf("\nThe Inorder display : ");
 INORDER(root);
printf("\nThe Preorder display : ");
 PREORDER(root);
printf("\nThe Postorder display : ");
POSTORDER(root);
}
break;
case 3: printf("\nenter the key to search:\n");
scanf("%d",&key);
SEARCH_NODE(root,key);
break;
case 4:  exit(0);
}
}
}
NODE  CREATE(NODE root,int data)
{
NODE newnode,x,parent;
newnode=(NODE)malloc(sizeof(struct node));
newnode->lchild=newnode->rchild=NULL;
newnode->info=data;
if(root==NULL)
root=newnode;
else
{
x=root;
while(x!=NULL)
{
parent=x;
if(x->info<data)
x=x->rchild; else if(x->info>data)
x=x->lchild;
else
{
printf("\nNode is already present in the tree\n");
 return(root);
}
```

```
}
if(parent->info<data)
parent->rchild=newnode;
else
parent->lchild=newnode;
}
return(root);
}

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

void PREORDER(NODE root)
{
if(root!=NULL)
{
printf("%d ",root->info);
PREORDER(root->lchild);
PREORDER(root->rchild);
}
}
void POSTORDER(NODE root)
{
if(root!=NULL)
{
POSTORDER(root->lchild);
POSTORDER(root->rchild);
printf("%d ",root->info);
}
}
NODE SEARCH_NODE(NODE root, int key)
{
```

```
NODE cur,q,parent,successor;
if(root==NULL)
{
printf("\nTree is empty\n");
 return root;
}
parent=NULL,cur=root;
while(cur!=NULL)
{
if(key==cur->info)
 break;
parent=cur;
cur= (key<cur->info)?cur->lchild:cur->rchild;
}
if(cur==NULL)
{
printf("\nData is not found\n");
return root;
}
printf("\nData %d is found\n",key);
}
```

Output:

1:CREATE
2:TREE TRAVERSAL
3.SEARCH
4.EXIT
Enter your choice
2

EMPTY TREE

1:CREATE
2:TREE TRAVERSAL
3.SEARCH
4.EXIT
Enter your choice
1

Enter data to be inserted
6

1:CREATE
2:TREE TRAVERSAL
3.SEARCH
4.EXIT
Enter your choice
1

Enter data to be inserted
9

1:CREATE
2:TREE TRAVERSAL
3.SEARCH
4.EXIT
Enter your choice
1

Enter data to be inserted
5

1:CREATE
2:TREE TRAVERSAL
3.SEARCH
4.EXIT
Enter your choice
1

Enter data to be inserted

2

1:CREATE
2:TREE TRAVERSAL
3.SEARCH
4.EXIT
Enter your choice
1

Enter data to be inserted
8

1:CREATE
2:TREE TRAVERSAL
3.SEARCH
4.EXIT
Enter your choice
1

Enter data to be inserted
15

1:CREATE
2:TREE TRAVERSAL
3.SEARCH
4.EXIT
Enter your choice
1

Enter data to be inserted
24

1:CREATE
2:TREE TRAVERSAL
3.SEARCH
4.EXIT
Enter your choice
1

Enter data to be inserted
14

1:CREATE
2:TREE TRAVERSAL
3.SEARCH
4.EXIT
Enter your choice
1

Enter data to be inserted
7

1:CREATE
2:TREE TRAVERSAL
3.SEARCH
4.EXIT
Enter your choice
1

Enter data to be inserted
8

Node is already present in the tree

1:CREATE
2:TREE TRAVERSAL
3.SEARCH
4.EXIT
Enter your choice
1

Enter data to be inserted
5

Node is already present in the tree

1:CREATE
2:TREE TRAVERSAL

3.SEARCH
4.EXIT
Enter your choice
2

The Inorder display : 2 5 6 7 8 9 14 15 24
The Preorder display : 6 5 2 9 8 7 15 14 24
The Postorder display : 2 5 7 8 14 24 15 9 6
1:CREATE
2:TREE TRAVERSAL
3.SEARCH
4.EXIT
Enter your choice
3

enter the key to search:
14

Data 14 is found

1:CREATE
2:TREE TRAVERSAL
3.SEARCH
4.EXIT
Enter your choice
3

enter the key to search:
4

Data is not found

Program 11:

Design, develop and implement a Program in C for the following operations on Graph (G) of Cities Create a Graph of N cities using Adjacency Matrix. Print all the nodes reachable from a given starting node in a digraph using BFS method

```
#include<stdio.h>
#include<stdlib.h>
 int n,a[10][10],i,j,source,s[10],choice,count;
void bfs(int n,int a[10][10],int source,int s[])  //BFS Algorithm
{
int q[10],u;
int front=1,rear=1;
s[source]=1;
q[rear]=source;
while(front<=rear)
{
u=q[front];
front=front+1;
for(i=1;i<=n;i++)
if(a[u][i]==1 && s[i]==0)
{
        rear=rear+1;
        q[rear]=i;
        s[i]=1;
}
}
}

int main()
{
printf("Enter the number of nodes : ");
scanf("%d",&n);
printf("\n Enter the adjacency matrix\n");
for(i=1;i<=n;i++)                  //Provide matrix of 0's and 1's
for(j=1;j<=n;j++)



scanf("%d",&a[i][j]);
```

```
while(1)
{
printf("\nEnter your choice\n");
printf("1.BFS\n 2.Exit\n");
scanf("%d",&choice);
switch(choice)
{
case 1: printf("\n Enter the source :");
scanf("%d",&source);            //Provide source for BFS
for(i=1;i<=n;i++)
s[i]=0;
bfs(n,a,source,s);
for(i=1;i<=n;i++)
{
if(s[i]==0)
printf("\n The node %d is not reachable",i);
else
printf("\n The node %d is reachable",i);
}
break;
case 2:
exit(0);
}
}
}
```

Output:

Enter the number of nodes : 5

Enter the adjacency matrix

0 1 0 1 0

0 0 1 0 0

1 0 0 0 0

0 0 1 0 0

0 0 1 1 0

Enter your choice

1.BFS

2.Exit

1

Enter the source :5

The node 1 is reachable

The node 2 is reachable

The node 3 is reachable

The node 4 is reachable

The node 5 is reachable

Enter your choice

1.BFS

2.Exit

1

Enter the source :1

The node 1 is reachable

The node 2 is reachable

The node 3 is reachable

The node 4 is reachable

The node 5 is not reachable

Enter your choice

1.BFS

2.Exit

2

PROGRAM 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>
#define MAX 5
#define mod(x) x%MAX
void linear_prob(int a[],int num,int key)
{
   if(a[key]==-1)
      a[key]=num;
   else
   {
      printf("\nCollision detected!!");
      int i;
      for(i=mod(key+1);i!=key;i=mod(++i))
         if(a[i]==-1)
         break;
      if(i!=key)
      {
         printf("\nCollision avoided successfully\n");
         a[i]=num;
      }
      else
         printf("\nHash table is full\n");
   }
}
void display(int a[])
{
   short ch,i;
   printf("\n1.Filtered display\n2.Display all\nEnter choice:");
   scanf("%hd",&ch);
```

```
    printf("\nHash table is :\n");
    for(i=0;i<MAX;i++)
        if(a[i]>0||ch-1)
            printf("%d  %d\n",i,a[i]);
}
int main()
{
    int a[MAX],num,i;
    printf("\nCollision handling by linear probing");
    for(i=0;i<MAX;a[i++]=-1);
    do
    {
        printf("\nEnter the data:");
        scanf("%4d",&num);
        linear_prob(a,num,mod(num));
        printf("Do u wish to continue(1/0):");
        scanf("%d",&i);
    }while(i);
    display(a);
    return 0;

}
```

Output:

```
Collision handling by linear probing
Enter the data:5
Do u wish to continue(1/0):1
Enter the data:4
Do u wish to continue(1/0):1
Enter the data:3
Do u wish to continue(1/0):0
1.Filtered display
2.Display all
Enter choice:1
Hash table is :
0 5
3  3
```

Appendix A

SAMPLE PROGRAMS ON DATA STRUCTURES

Program 1

Write a C program to implement pointer operations.

```
// Program to implement pointer operations
#include<stdio.h>
#include<conio.h
> void main()
{
    int i_num =3, *i_ptr=&i_num;
    clrscr();
    printf("\nAddress of i_num=0X%x",&i_num);
    printf("\nAddress of i_num=0X%x",i_ptr);
    printf("\nAddress of i_ptr=0X%x",&i_ptr);
    printf("\nAddressof i_num through pointer=0X%x",i_ptr);
    printf("\nValue of i_num=%d",i_num);
    printf("\nValue of i_num through pointer symbols=%d",*(&i_num));
    printf("\nValue of I through pointer=%d",*i_ptr);
    getch();
}
```

Program2

Write a C program to exchange two values using pointers.

```
// Program to exchange two values
#include<stdio.h>
#include<conio.h>
void main()
 {
      int  i_num1,  i_num2;
      void swap(int*, int*);
      clrscr();
      printf("Enter        2        integers\n");
      scanf("%d%d", &i_num1, &i_num2);
      printf("Values before swapping are %d and %d\n\n",i_num1,i_num2);
      swap(&i_num1,&i_num2);


            printf("Values after swapping are %d and %d\n\n",i_num1,i_num2);
      getch();
}

void swap(int *x,int *y)

{

      int temp =*x;
      *x=*y;
      *y=temp;

}
```

Program 3

Write a program to create a structure named student with variables book ID, name and price. Create a variable of type structure, initialize the details and print them on the screen.

// Program to implement structure

```
#include <stdio.h>
#include
<conio.h> struct
book
    {       int
        i_book
        _id;
        char
        *c_na
        me;
        float
        f_price
        ;
} b;
void main()

{

    clrscr();
    b.i_book_id
    =1;
    b.c_name = "Fundamentals of Data
    Structure"; b.f_price = 390.5;
    printf("Book Id is : %d \n",
    b.i_book_id); printf("Name is : %s
    \n", b.c_name); printf("Price is : %f
    \n", b.f_price); getch();
}
```

APPENDIX B:

VIVA QUESTIONS WITH

ANSWERS

1. What is data structure?

A data structure is a way of organizing data that considers not only the items stored, but also their relationship to each other.

2. 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

3. What are the major data structures used in the following areas: RDBMS, Network data model and Hierarchical data model?

- RDBMS = Array (i.e. Array of structures)
- Network data model = Graph
- Hierarchical data model = Trees

4. What is the 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.

5. 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.

6. In tree construction which is the suitable efficient data structure? (Array, Linked list, Stack, Queue)
Linked list is the suitable efficient data structure.

7. What is the significant difference between ARRAY and STACK?

Stack uses LIFO. Thus the item that is certainly first entered would be the last removed. In array the items could be entered or removed in any order. Basically each and every member access is done making use of index and no strict order is to be followed here to remove a particular element. Array could be multi-dimensional or one dimensional but stack really should be one-dimensional. Size of array is fixed, while stack may be grow or shrink. We can say stack is actually dynamic data structure.

8. Explain whether Linked List is actually linear or Non-linear data structure? Link list is definitely obviously linear data structure simply because each and every element (NODE) acquiring specific place and as well each and every component has got its unique successor in addition to predecessor.

9. What is the difference between a Stack and a Queue?

Stack follows LIFO (Last In First Out) method of accessing the members. Queue follows FIFO (First In First Out) method of accessing the members.

10. What is a one way chain or singly linked linear list?

Singly linked list is a collection of nodes. Each node has element and address of the next element. With that address only forward traversal is possible i.e. single link

11. How can a node be inserted in the middle of a linked list?

By repointing the previous and the next elements of existing nodes to the new node.

11. What is the heap?

The heap is where malloc(), calloc(), and realloc() get memory.

12. Which data structure is needed to convert infix notations to post fix notations?

Stack

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

- Sparse matrix.
- Index generation.

14. In tree construction which is the suitable efficient data structure?

Linked List

15. What is the evaluation order according to which an infix expression is converted to postfix expression?

The evaluation order is:

- Brackets or Parenthesis,
- Exponentiation,
- Multiplication or division,
- Addition or subtraction.

16. Give the advantages of using post fix notations over infix notations?

An infix expression is evaluated keeping in mind the precedence of operators. An infix expression is difficult for the machine to know and keep track of precedence of operators. A postfix expression itself takes care of the precedence of operators as the placement of operators. Thus, for the machine, it is easier to carry out a postfix expression than an infix expression.

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

The manipulation of Arithmetic expression, Symbol Table construction, Syntax analysis.

18. 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.

19. Why do we Use a Multidimensional Array?

A multidimensional array can be useful to organize subgroups of data within an array.

20. Run Time Memory Allocation is known as?

Allocating memory at runtime is called a dynamically allocating memory. In this, you dynamically allocate memory by using the new operator when declaring the array, for example: int grades [] = new int [10];