



“DATA STRUCTURES LAB”

Subcode: 18CSL38

Prepared By

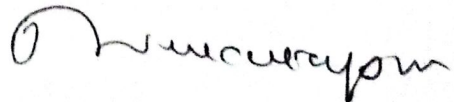
Dr. K. Venkata Rao
Professor

Dr. Vijayalaxmi Mekali
Associate Professor

Mr. Raghavendrachar S
Assistant Professor

Dr. Vaneeta M
Associate Professor

Department of Computer Science & Engineering
K.S Institute of Technology,
Bangalore-109.


Head of the Department
Dept. of Computer Science & Engg
K.S. Institute of Technology
Bangaluru -560 109



K.S. INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

Vision of the Institute

To impart quality technical education with ethical values, employable skills and research to achieve excellence

Mission of the Institute

- To attract and retain highly qualified, experienced & committed faculty.
- To create relevant infrastructure.
- Network with industry & premier institutions to encourage emergence of new ideas by providing research & development facilities to strive for academic excellence.
- To inculcate the professional & ethical values among young students with employable skills & knowledge acquired to transform the society.

Vision of the Department

To create competent professionals in Computer Science and Engineering with adequate skills to drive the IT industry

Mission of the Department

- Impart sound technical knowledge and quest for continuous learning.
- To equip students to furnish Computer Applications for the society through experiential learning and research with professional ethics.
- Encourage team work through inter-disciplinary project and evolve as leaders with social concerns.

Head of the Department
Dept. of Computer Science & Engg.
K.S. Institute of Technology
Bengaluru -560 109



K.S. INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

Program Educational Objectives

- PEO1:** Excel in professional career by acquiring knowledge in cutting edge technology and contribute to the society as an excellent employee or as an entrepreneur in the field of Computer Science & Engineering.
- PEO2:** Continuously enhance their knowledge on par with the development in IT industry and pursue higher studies in Computer Science & Engineering.
- PEO3:** Exhibit professionalism, cultural awareness, team work, ethics, and effective communication skills with their knowledge in solving social and environmental problems by applying computer technology.

Program Specific Outcomes (PSO)

- PSO1:** Ability to understand, analyze problems and implement solutions in programming languages, as well to apply concepts in core areas of Computer Science in association with professional bodies and clubs.
- PSO2:** Ability to use computational skills and apply software knowledge to develop effective solutions and data to address real world challenges.

D. Weerap

Head of the Department
Dept. of Computer Science & Engg.
K.S. Institute of Technology
Bengaluru - 560 100



K.S. INSTITUTE OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

Program Outcomes

- PO1: Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
- PO2: Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
- PO3: Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
- PO4: Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
- PO5: Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
- PO6: The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
- PO7: Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

- PO8: Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
- PO9: Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- PO10: Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
- PO11: Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
- PO12: Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.



Head of the Department
Dept. of Computer Science & Engg
K.S. Institute of Technology
Bengaluru - 560 103



K. S. INSTITUTE OF TECHNOLOGY

#14, Raghuvanahalli, Kanakapura Main Road, Bengaluru-5600109

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Course: Data Structures Laboratory			
Type: Core		Course Code:18CSL38	
No of Hours per week			
Number of Lecture Hours/Week		Total/Week	Total teaching hours
01I + 02P		3hrs	33
Marks			
Internal Assessment	Examination	Total	Credits
40	60	100	2
<u>Aim/Objective of the Course:</u>			
To study the different types of data structures, operations on data structures and their applications			
Course Learning Outcomes:			
After completing the course, the students will be able to,			Bloom's Level
18CSL38.1	Select implementation procedures to demonstrate the various operations on different data structures.		Applying (K3)
18CSL38.2	Develop the C programs to implement various operations on different data structures.		Applying (K3)
18CSL38.3	Identity suitable data structures to implement stack, queues, trees, graphs and hash table.		Applying (K3)
18CSL38.4	Experiment with C programs to implement stack, queues, trees, graph and hash table with different data structures.		Applying (K3)
18CSL38.5	Apply concepts of arrays, stack, queue, linked list, trees and graph to solve different problems.		Applying (K3)
Syllabus Content			
1. Design, Develop and Implement a menu driven Program in C for the following Array operations a. Creating an Array of N Integer Elements b. Display of Array Elements with Suitable Headings c. Inserting an Element (ELEM) at a given valid Position (POS) d. Deleting an Element at a given valid Position(POS) e. Exit. Support the program with functions for each of the above operations.			CO1,CO2 CO3,CO4 CO5 PO1-3, PO2-2 PO3-2, PO5-1 PSO1-2, PSO2-2

2. Design, Develop and Implement 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.	CO1,CO2 CO3,CO4 CO5 PO1-3, PO2-2 PO3-2, PO5-1 PSO1-2, PSO2-2
3. Design, Develop and Implement 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 Support the program with appropriate functions for each of the above operations	CO1,CO2 CO3,CO4 CO5 PO1-3, PO2-2 PO3-2, PO5-1 PSO1-2, PSO2-2
4. Design, Develop and Implement 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.	CO1,CO2 CO3,CO4 CO5 PO1-3, PO2-2 PO3-2, PO5-1 PSO1-2, PSO2-2
5. Design, Develop and Implement 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 b. Solving Tower of Hanoi problem with n disks	CO1,CO2 CO3,CO4 CO5 PO1-3, PO2-2 PO3-2, PO5-1 PSO1-2, PSO2-2
6. Design, Develop and Implement 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 Continued	CO1,CO2 CO3,CO4 CO5 PO1-3, PO2-2 PO3-2, PO5-1 PSO1-2, PSO2-2
7. Design, Develop and Implement a menu driven Program in C for the following operations on Singly Linked List (SLL) of Student Data with the fields: USN, Name, Branch, 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	CO1,CO2 CO3,CO4 CO5 PO1-3, PO2-2 PO3-2, PO5-1 PSO1-2, PSO2-2
8. Design, Develop and Implement 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	CO1,CO2 CO3,CO4 CO5 PO1-3, PO2-2 PO3-2, PO5-1 PSO1-2, PSO2-2

9. Design, Develop and Implement 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	CO1,CO2 CO3,CO4 CO5 PO1-3, PO2-2 PO3-2, PO5-1 PSO1-2, PSO2-2
10. Design, Develop and Implement 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 e. Exit	CO1,CO2 CO3,CO4 CO5 PO1-3, PO2-2 PO3-2, PO5-1 PSO1-2, PSO2-2
11. Design, Develop and Implement 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	CO1,CO2 CO3,CO4 CO5 PO1-3, PO2-2 PO3-2, PO5-1 PSO1-2, PSO2-2
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. Design and develop a Program in C that uses Hash function H: K L as $H(K)=K \text{ mod } m$ (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	CO1,CO2 CO3,CO4 CO5 PO1-3, PO2-2 PO3-2, PO5-1 PSO1-2, PSO2-2
Assessment: - 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 (Courseed to change in accordance with university regulations) c) For laboratories having only one part – Procedure + Execution + Viva-Voce: 15+70+15 = 100 Marks	

CO to PO Mapping**PO1:** Science and engineering Knowledge**PO2:** Problem Analysis**PO3:** Design & Development**PO4:** Investigations of Complex Problems**PO5:** Modern Tool Usage**PO6:** Engineer & Society**PO7:** Environment and Society**PO8:** Ethics**PO9:** Individual & Team Work**PO10:** Communication**PO11:** Project Management & Finance**PO12:** Life long Learning

PSO1: Ability to understand, analyze problems and implement solutions in programming languages, as well to apply concepts in core areas of Computer Science in association with professional bodies and clubs.

PSO2: Ability to use computational skills and apply software knowledge to develop effective solutions and data to address real world challenges.

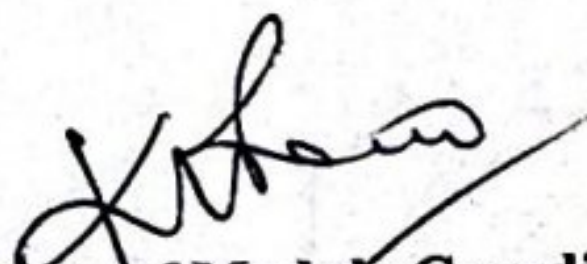
CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
18CSL38.1	3	2	2		1							
18CSL38.2	3	2	2		1							
18CSL38.3	3	2	2		1							
18CSL38.4	3	2	2		1							
18CSL38.5	3	2	2		1							
18CSL38	3	2	2		1							

CO	PSO1	PSO2
18CSL38.1	2	2
18CSL38.2	2	2
18CSL38.3	2	2
18CSL38.4	2	2
18CSL38.5	2	2
18CSL38	2	2

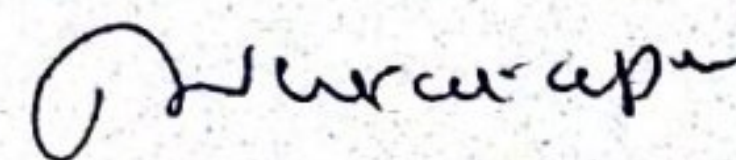
3	Substantial (High) Correlation
2	Moderate (Medium) Correlation
1	Slight (Low) Correlation
-	No correlation.



Signature of Course In-Charge



Signature of Module Coordinator



Signature of HOD

Head of the Department
 Dept. of Computer Science & Engg.
 K.S. Institute of Technology
 Bengaluru - 560 009

RUBRICS FOR LABORATORY EVALUATION				
	Exceptional(10-9)	Acceptable(8-6)	Marginal(5-1)	Unsatisfactory(0)
Record (10)	The Record is well written and clearly explains what the experiments are accomplishing	The record is written and useful in understanding	The record is vague only the reader can understand.	The record does not help the reader understand
Program Logic (In the Observation book)(10)	The program executes and meets all of the specifications.	The Program executes correctly with no errors and meets few of the specifications.	The program executes with few logical errors	Program does not execute
Viva(10)	Demonstrates deep knowledge; answer the questions with explanations and elaboration.	Adequate knowledge of most topics; answer the questions, but fails to elaborate.	Superficial knowledge of topic; only able to answer basic questions.	Not able to answer basic questions
Internal Test (10)	Write up of the program : 02			
	Execution of the Program : 06			
	Viva: 02			

DATA STRUCTURES LABORATORY (Effective from the academic year 2018 -2019) SEMESTER – III			
Course Code	18CSL38	CIE Marks	40
Number of Contact Hours/Week	0:2:2	SEE Marks	60
Total Number of Lab Contact Hours	36	Exam Hours	03
Credits – 2			
Course Learning Objectives: This course (18CSL38) will enable students to: This laboratory course enable students to get practical experience in design, develop, implement, analyze and evaluation/testing of - Asymptotic performance of algorithms. - Linear data structures and their applications such as stacks, queues and lists - Non-Linear data structures and their applications such as trees and graphs - Sorting and searching algorithms			
Descriptions (if any): Implement all the programs in ‘C / C++’ Programming Language and Linux / Windows as OS.			
Programs List:			
1.	Design, Develop and Implement a menu driven Program in C for the following array operations. - Creating an array of N Integer Elements - Display of array Elements with Suitable Headings - Inserting an Element (ELEM) at a given valid Position (POS) - Deleting an Element at a given valid Position (POS) - Exit. Support the program with functions for each of the above operations.		
2.	Design, Develop and Implement a Program in C for the following operations on Strings. - Read a main String (STR), a Pattern String (PAT) and a Replace String (REP) - 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.		
3.	Design, Develop and Implement a menu driven Program in C for the following operations on STACK of Integers (Array Implementation of Stack with maximum size MAX) - Push an Element on to Stack - Pop an Element from Stack - Demonstrate how Stack can be used to check Palindrome - Demonstrate Overflow and Underflow situations on Stack - Display the status of Stack - Exit Support the program with appropriate functions for each of the above operations		
4.	Design, Develop and Implement 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.		
5.	Design, Develop and Implement a Program in C for the following Stack Applications - Evaluation of Suffix expression with single digit operands and operators: +, -, *, /, %, ^ - Solving Tower of Hanoi problem with n disks		

6.	Design, Develop and Implement a menu driven Program in C for the following operations on Circular QUEUE of Characters (Array Implementation of Queue with maximum size MAX) - Insert an Element on to Circular QUEUE - Delete an Element from Circular QUEUE - Demonstrate Overflow and Underflow situations on Circular QUEUE - Display the status of Circular QUEUE - Exit Support the program with appropriate functions for each of the above operations
7.	Design, Develop and Implement a menu driven Program in C for the following operations on Singly Linked List (SLL) of Student Data with the fields: <i>USN, Name, Programme, Sem, PhNo</i> - Create a SLL of N Students Data by using <i>front insertion</i>. - Display the status of SLL and count the number of nodes in it - Perform Insertion / Deletion at End of SLL - Perform Insertion / Deletion at Front of SLL(Demonstration of stack) - Exit
8.	Design, Develop and Implement a menu driven Program in C for the following operations on Doubly Linked List (DLL) of Employee Data with the fields: <i>SSN, Name, Dept, Designation, Sal, PhNo</i> - Create a DLL of N Employees Data by using <i>end insertion</i>. - Display the status of DLL and count the number of nodes in it - Perform Insertion and Deletion at End of DLL - Perform Insertion and Deletion at Front of DLL - Demonstrate how this DLL can be used as Double Ended Queue. - Exit
9.	Design, Develop and Implement a Program in C for the following operations on Singly Circular Linked List (SCLL) with header nodes - Represent and Evaluate a Polynomial $P(x,y,z) = 6x^2y^2z - 4yz^5 + 3x^3yz + 2xy^5z - 2xyz^3$ - 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
10.	Design, Develop and Implement a menu driven Program in C for the following operations on Binary Search Tree (BST) of Integers . - Create a BST of N Integers: 6, 9, 5, 2, 8, 15, 24, 14, 7, 8, 5, 2 - Traverse the BST in Inorder, Preorder and Post Order - Search the BST for a given element (KEY) and report the appropriate message - Exit
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 DFS/BFS method
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. Design and develop a Program in C that uses Hash function $H: K \rightarrow L$ as $H(K) = K \bmod m$ (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.
Laboratory Outcomes: The student should be able to:	

- Analyze and Compare various linear and non-linear data structures
- Code, debug and demonstrate the working nature of different types of data structures and their applications
- Implement, analyze and evaluate the searching and sorting algorithms
- Choose 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 (*Courseed to change in accordance with university regulations*)
 - c) For laboratories having only one part – Procedure + Execution + Viva-Voce: 15+70+15 = 100 Marks
 - d) For laboratories having PART A and PART B
 - i. Part A – Procedure + Execution + Viva = 6 + 28 + 6 = 40 Marks
 - ii. Part B – Procedure + Execution + Viva = 9 + 42 + 9 = 60 Marks

PROGRAM 1.

1. Design, Develop and Implement a menu driven Program in C for the following Array operations

- a. Creating an Array of N Integer Elements**
- b. Display of Array Elements with Suitable Headings**
- c. Inserting an Element (ELEM) at a given valid Position (POS)**
- d. Deleting an Element at a given valid Position(POS) e. Exit.**

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

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 10
int a[MAX],n;
void create(),display(),insert(),delete();
void main()
{
    int ch;
    printf("\n IMPLEMENTATION OF OPERATIONS ON ARRAY\n");
    for(;;)
    {
        printf("\nMENU\n");
        printf("\n1:CREATE \n2:DISPLAY\n3:INSERT\n4:DELETE\n");
        printf("\nEnter your choice::");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:create();
                break;
            case 2:display();
                break;
            case 3:insert();
                break;
            case 4:delete();
                break;
            case 5:exit(0);
        }
        /*end of switch */
    } /*end of for */
} /*end of main */
```

```
void create()
{
    int i;
    printf("\nARRAY CREATION\n");
    printf("\nEnter the number of elements in array::");
    scanf("%d",&n);
    printf("\nEnter the elements\n");
    for(i=0;i<n;i++)
        scanf("%d",&a[i]);
}
void display()
{
    int i;
    printf("\n ARRAY ELEMENTS ARE::\n");
    for(i=0;i<n;i++)
        printf("%d\t", a[i]);
}
void insert()
{
    int i,ele,pos;
    printf("\nELEMENT INSERTION\n");
    if(n==MAX)
        printf("\n ARRAY IS FULL, INSERSION IS NOT POSSIBLE\n");
    else
    {
        printf("\n Enter the position less or equal to n+1 to insert the  element::");
        scanf("%d",&pos);
        if(pos<=n+1)
        {
            printf("\nEnter the element to insert::");
            scanf("%d",&ele);
            for(i=n-1;i>=pos-1;i--)
                a[i+1]=a[i];
            a[pos-1]=ele;
            ++n;
        } /*end of if*/
        else
            printf("\nInvalid position\n");
    } /*end of else*/
} /*end of insert function*/
```

```
void delete()
{
    int item,i,pos;
    printf("\nELEMENT DELETION\n");
    if(n==0)
        printf("\nARRAY IS EMPTY DELETION NOT POSSIBLE\n");
    else
    {
        printf("\nEnter the position less than or equal to n to delete element::");
        scanf("%d",&pos);
        if(pos<=n)
        {
            item=a[pos-1];
            printf("\nDELETED ITEM IS::%d\n",item);
            --n;
            for(i=pos-1;i<n;i++)
                a[i]=a[i+1];
        }/*end of if*/
        else
            printf("\nInvalid position\n");
    }/*end of else*/
}/*end of delete function*/
```

OUTPUT

IMPLEMENTATION OF OPERATIONS ON ARRAY

MENU

1:CREATE
2:DISPLAY
3:INSERT
4:DELETE

Enter your choice::1

ARRAY CREATION

Enter the number of elements in array::5

Enter the elements

12 13 14 15 16

MENU

1:CREATE
2:DISPLAY
3:INSERT
4:DELETE

Enter your choice::2

ARRAY ELEMENTS ARE::

12 13 14 15 16

MENU

1:CREATE
2:DISPLAY
3:INSERT
4:DELETE

Enter your choice::3

ELEMENT INSERTION

Enter the position less or equal to n+1 to insert the element::4

Enter the element to insert::100

MENU

1:CREATE
2:DISPLAY

3:INSERT

4:DELETE

Enter your choice::2

ARRAY ELEMENTS ARE::

12 13 14 100 15 16

MENU

1:CREATE

2:DISPLAY

3:INSERT

4:DELETE

Enter your choice::3

ELEMENT INSERTION

Enter the position less or equal to n+1 to insert the element::9

Invalid position

MENU

1:CREATE

2:DISPLAY

3:INSERT

4:DELETE

Enter your choice::4

ELEMENT DELETION

Enter the position less than or equal to n to delete element::7

Invalid position

MENU

1:CREATE

2:DISPLAY

3:INSERT

4:DELETE

Enter your choice::4

ELEMENT DELETION

Enter the position less than or equal to n to delete element::4

DELETED ITEM IS::100

MENU

1:CREATE
2:DISPLAY
3:INSERT
4:DELETE

Enter your choice::2

ARRAY ELEMENTS ARE::
12 13 14 15 16

PROGRAM 2.

2 Design, Develop and Implement 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>
void read(char str[],char pat[],char rep[])
{
    printf("\nEnter a string:: \n");
    gets(str);
    printf("\nEnter a pattern string:: \n");
    gets(pat);
    printf("\nEnter a replace string ::\n");
    gets(rep);
}
void patreplace(char str[],char pat[],char rep[],char ans[])
{
    int i,m,c,j,count,k;
    i=m=c=j=count=0;
    while ( str[c] != '\0')
    {
        if ( str[m] == pat[i] )
        {
            i++;
            m++;
            if ( pat[i] == '\0')
            {
                count++;
                for(k=0; rep[k] != '\0';k++,j++)
                    ans[j] = rep[k];
                i=0;
                c=m;
            }/*end of inner if*/
        }/*end of outer if*/
        else
        {
            ans[j] = str[c];
            j++;
            c++;
            m = c;
            i=0;
        }/*end of else*/
    }/*end of while*/
}
```

```
ans[j] = '\0';
if(count==0)
    printf("No pattern in the string\n");
else
    printf("\nThe resultant string is\n%s" ,ans);
}/*end of patternreplace function*/

void main()
{
char str[100],pat[100],rep[100],ans[100];
printf("Read main string, pattern string and replace string\n");
read(str,pat,rep);
patreplace(str,pat,rep,ans);
}
```

OUTPUT

Read main string, pattern string and replace string

O/P1

Enter a string::

KSITRNSIT

Enter a pattern string::

SIT

Enter a replace string ::

AAAAA

The resultant string is

KAAAAARNAAAAA

Read main string, pattern string and replace string

O/P1

Enter a string::

KSITRNSIT

Enter a pattern string::

IIT

Enter a replace string ::

BBBBB

No pattern in the string

PROGRAM 3.

3 Design, Develop and Implement 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

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define max 5
void push(),pop(),display(),overflow_underflow(),palindrome();
int stack[max],top=-1,top1=-1;
void main()
{
    int ch,item;
    printf("\nStack Operations\n");
    for(;;)
    {
        printf("\n1.Push\n2.Pop\n3.Character Stack to check palindrome\n
               4.Overflow_Underflow\n5.Display\n6.Exit\n");
        printf("\nEnter your Choice:");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:push();
                    break;
            case 2: pop();
                    break;
            case 3:palindrome();
                    break;
            case 4:overflow_underflow();
                    break;
            case 5:display();
                    break;
            case 6: exit(0);
        }
        /*end of switch*/
    }
    /*end of for*/
} /* end of main*/
```

```
void push()
{
    int ele;
    if(top==max-1)
        printf("\nstack overflow\n");
    else
    {
        printf("\nEnter element to push:");
        scanf("%d",&ele);
        stack[++top]=ele;
    } /*end of else*/
} /*end of push function*/

void pop()
{
    int num;
    if(top== -1)
        printf("\nUnderflow\n");
    else
    {
        num=stack[top--];
        printf("\nPoped element::%d\n",num);
    } /*end of else*/
} /* end of pop function*/

void overflow_underflow()
{
    if(top==max-1)
        printf("\nStack full\n");
    if(top== -1)
        printf("\nStack underflow\n");
}

void display()
{
    int i;
    if(top== -1)
        printf("\nStack is Empty\n");
    else
    {
        printf("\n Elements on stack are::\n");
        for(i=top;i>=0;i--)
            printf("%d\n",stack[i]);
    } /*end of else*/
} /*end of display function*/
```

```
void palindrome()
{
    int i=0,j=0;
    char s[10],st[10],rev[10];
    printf("\nEnter the string to check palindrome::");
    scanf("%s",s);
    while(s[i]!='\0')
    {
        st[++top1]=s[i];
        i++;
    }/* end of while*/
    while(top1!=-1)
    {
        rev[j]=st[top1--];
        j++;
    }/* end of while*/
    rev[j]='\0';
    if(strcmp(s,rev)==0)
        printf("\nString %s is palindrome\n",s);
    else
        printf("\nString %s is not palindrome\n",s);
}/* end of palindrome function*/
```

OUTPUT

Stack Operations

- 1.Push
- 2.Pop
- 3.Character Stack to check palindrom
- 4.Overflow_Underflow
- 5.Display
- 6:Exit

Enter your Choice:1

Enter element to push:10

- 1.Push
- 2.Pop
- 3.Character Stack to check palindrome
- 4.Overflow_Underflow
- 5.Display
- 6:Exit

Enter your Choice:1

Enter element to push:20

- 1.Push
- 2.Pop
- 3.Character Stack to check palindrome
- 4.Overflow_Underflow
- 5.Display
- 6.Exit

Enter your Choice:1

Enter element to push:30

- 1.Push
- 2.Pop
- 3.Character Stack to check palindrome
- 4.Overflow_Underflow
- 5.Display
- 6.Exit

Enter your Choice:1

Enter element to push:40

- 1.Push
- 2.Pop
- 3.Character Stack to check palindrome
- 4.Overflow_Underflow
- 5.Display
- 6.Exit

Enter your Choice:1

Enter element to push:50

- 1.Push
- 2.Pop
- 3.Character Stack to check palindrome
- 4.Overflow_Underflow
- 5.Display
- 6.Exit

Enter your Choice:1

stack overflow

- 1.Push
- 2.Pop
- 3.Character Stack to check palindrome
- 4.Overflow_Underflow
- 5.Display
- 6:Exit

Enter your Choice:5

Elements on stack are::

50
40
30
20
10

- 1.Push
- 2.Pop
- 3.Character Stack to check palindrome
- 4.Overflow_Underflow
- 5.Display
- 6:Exit

Enter your Choice:2

Poped element::50

- 1.Push
- 2.Pop
- 3.Character Stack to check palindrome
- 4.Overflow_Underflow
- 5.Display
- 6:Exit

Enter your Choice:2

Poped element::40

- 1.Push
- 2.Pop
- 3.Character Stack to check palindrome
- 4.Overflow_Underflow
- 5.Display
- 6:Exit

Enter your Choice:5

Elements on stack are::

30

20

10

1.Push

2.Pop

3.Character Stack to check palindrome

4.Overflow_Underflow

5.Display

6:Exit

Enter your Choice:3

Enter the string to check palindrom::bharath

String bharath is not palindrom

1.Push

2.Pop

3.Character Stack to check palindrome

4.Overflow_Underflow

5.Display

6:Exit

Enter your Choice:3

Enter the string to check palindrome::dad

String dad is palindrome

1.Push

2.Pop

3.Character Stack to check palindrome

4.Overflow_Underflow

5.Display

6:Exit

Enter your Choice:2

Poped element::30

1.Push

- 2.Pop
- 3.Character Stack to check palindrome
- 4.Overflow_Underflow
- 5.Display
- 6:Exit

Enter your Choice:2

Poped element::20

- 1.Push
- 2.Pop
- 3.Character Stack to check palindrome
- 4.Overflow_Underflow
- 5.Display
- 6:Exit

Enter your Choice:2

Poped element::10

- 1.Push
- 2.Pop
- 3.Character Stack to check palindrome
- 4.Overflow_Underflow
- 5.Display
- 6:Exit

Enter your Choice:4

Stack underflow

- 1.Push
- 2.Pop
- 3.Character Stack to check palindrome
- 4.Overflow_Underflow
- 5.Display
- 6:Exit

Enter your Choice:

PROGRAM 4:

4 Design, Develop and Implement 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>
```

```
typedef enum{lparen,rparen,plus,minus,mul,div,mod,pow,eos,op}pred;
```

/* enum data pred gives value to (,), +,-,*, /, %,^ lparen is 0, rparen is 1, plus is 2 Minus is 3, mul is 4, div is 5, mod is 6, pow is 7, eos is 8, op(operand) is 9 program uses these value ex: in place of lparen it uses 0, ex: in place of plus it uses 2 so on isp stands for In Stack Priority and icp stands for In Coming Priority*/

```
int isp[ ]={0,19,12,12,13,13,13,14,0}; /* priority of operator when it is in stack*/
```

```
int icp[ ]={20,19,12,12,13,13,13,15,0}; /* priority of operator when it is out of stack*/
```

```
char exp[50],symbol; /* infix expression*/
```

```
pred stack[50]; /* stack contains only value of type enum pred*/
```

```
int top=0,n=0;
```

```
pred readtoken() /* to read character from infix expression*/
```

```
{
```

```
symbol=exp[n++];
```

```
switch(symbol)
```

```
{
```

```
    case '(':return lparen;
```

```
    case ')':return rparen;
```

```
    case '+':return plus;
```

```
    case '-': return minus;
```

```
    case '*':return mul;
```

```
    case '/':return div;
```

```
    case '%':return mod;
```

```
    case '^':return pow;
```

```
    case '\0':return eos;
```

```
    default:return op;
```

```
}
```

```
}
```

```
void displaytoken(pred token)
```

```
{
```

```
switch(token)
```

```
{
```

```
    case plus:printf("+");
```

```
        break;
```

```
        case minus:printf("-");
                    break;
        case mul:printf("*");
                    break;
        case div:printf("/");
                    break;
        case mod:printf("%");
                    break;
        case pow:printf("^");
                    break;
    }
}
void push(pred item)
{
    stack[++top]=item;
}
pred pop()
{
    return stack[top--];
}
void infixtopost()
{
    pred token,token1;
    stack[0]=eos;
    token=readtoken();
    while(token!=eos) /* while loop to read the token*/
    {
        if(token==op) /* if character is operand copy to postfix*/
        {
            printf("%c",symbol);
        }
        /*if character is right parenthesis pop stack until left paranthesis on stack*/
        else if(token==rparen)
        {
            while(stack[top]!=lparen)
            {
                token1=pop();
                displaytoken(token1);
            }
            pop(); /*pop left paranthesis from stack and ignore it*/
        }
    }
}
```

```
        else
        {
            while(isp[stack[top]]>=icp[token])
/*if priority of operator on stack is > =priority of incoming operator pop stack until operator with
less priority comes on stack */
            {
                token1=pop();
                displaytoken(token1);
            } /* copy popped character to postfix ie display*/
            push(token);/* push incoming character to stack*/
        }

token=readtoken(symbol,n);
}
/*once end of infix expression if any operators remains on stack, pop all operators copy to
postfix ie display
while((token=pop())!=eos)
{
    displaytoken(token);
}
printf("\n");
}

void main()
{
    printf("PROGRAM TO CONVER INFIX EXP TO POSTFIX EXP\n\n");
    printf("Enter infix expression:");
    gets(exp);
    printf("Postfix expression:");
    infixtopost();
}
```

OUTPUT

PROGRAM TO CONVER INFIX EXP TO POSTFIX EXP

Enter infix expression:a^(b+c%d)+e
Postfix expression:abcd%+^e+

PROGRAM TO CONVER INFIX EXP TO POSTFIX EXP

Enter infix expression:a+(b*c-(d/e)+f)
Postfix expression: abc*de/-f++

PROGRAM 5.**5 Design, Develop and Implement 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

a. Evaluation of Suffix expression

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
#include<math.h>
#include<stdlib.h>
#define STACK_SIZE 50
/* isnum() function checks whether a character is an alphanumeric character(a-z or A-Z,0-9)*/
/* isdigit() function checks whether character is numeric character(0-9) or not*/
/* atoi() converts string to integer*/

int stack[STACK_SIZE];
int top = -1;

void push(int a)
{
    stack[++top] = a;
    return;
}
int pop()
{
    return (stack[top--]);
}

int eval(char s[])
{
    int i;
    int temp, op1, op2, result;
    char symbol[1];
    for(i = 0; i < strlen(s); i++)
    {
        if(isalnum(s[i]))
        {
            if(isdigit(s[i]))
            {
                symbol[0] = s[i];
                temp = atoi(symbol);
                push(temp);
            }/*end of inner if*
        }
    }
}
```

```
        else
        {
            printf("\n\nEnter a value of %c::",s[i]);
            scanf("%d", &temp);
            push(temp);
        }/*end of else of inner if*/
    }/* end of outer if */
    else
    {
        op2 = pop();
        op1 = pop();
        switch(s[i])
        {
            case '^': result = pow(op1, op2);
                      break;
            case '*': result = op1 * op2;
                      break;
            case '/': if(op2==0)
                      {
                          printf("Divide by zero error\n\n");
                          exit(0);
                      }/*end of if*/
            else
                result = op1 / op2;
                      break;
            case '%': if(op2==0)
                      {
                          printf("Divide by zero error\n\n");
                          exit(0);
                      }
            else
                result = op1 %op2;
                      break;
            case '+': result = op1 + op2;
                      break;
            case '-': result = op1 - op2;
                      break;
            default:
                printf("Invalid Expression\n\n");
                exit(0);
        }/*end of switch*/
        push(result);
    }/*end of else*/
}/*end of for*/
result = pop();
```

```
if(top != -1)
{
    printf("Invalid Expression\n");
    exit(0);
}
return result;

}/*end of eval function */

void main()
{
    char s[50];
    printf("\t\tEVALUATION OF POSTFIX EXPRESSION\n");
    printf("\n\nEnter the postfix expression\n");
    gets(s);
    printf("\n\nThe result of the evaluation is: %d\n", eval(s));
    return;
}
```

Note: To Compile the 5.a program use command (gcc filename.c -lm)

OUTPUT

EVALUATION OF POSTFIX EXPRESSION

Enter the postfix expression

ab*cd/+

Enter a value of a::2

Enter a value of b::3

Enter a value of c::8

Enter a value of d::4

The result of the evaluation is: 8

EVALUATION OF POSTFIX EXPRESSION

Enter the postfix expression

ab%cd^+e+

Enter a value of a::9

Enter a value of b::4

Enter a value of c::2

Enter a value of d::3

Enter a value of e::9

The result of the evaluation is: 18

EVALUATION OF POSTFIX EXPRESSION

Enter the postfix expression

93%24^+

The result of the evaluation is: 16

5b. Solving Tower of Hanoi problem with n disks

```
#include<stdio.h>
int tower(int n,char s, char t, char d)
{
    if(n==0)
        return 0;
    tower(n-1,s,d,t);
    printf("\n Move disk %d from %c to %c\n",n,s,d);
    tower(n-1,t,s,d);
    return 0;
}
void main()
{
    char A,B,C;
    int n;
    printf("Enter the number of elements\n\n");
    scanf("%d",&n);
    printf("\n Sequence of disk\n");
    tower(n,'A','B','C');
    printf("\n");
}
```

OUTPUT:

1.Enter the number of disks:3

Sequence of disk

Move disk 1 from A to C

Move disk 2 from A to B

Move disk 1 from C to B

Move disk 3 from A to C

Move disk 1 from B to A

Move disk 2 from B to C

Move disk 1 from A to C

PROGRAM 6.

6. Design, Develop and Implement 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

```
#include <stdio.h>
#include<stdlib.h>
#include<stdio_ext.h>
#define max 5
int front=0,rear=-1,count=0;
char circularq[max];
void insert(),delete(),display();
void main()
{
    int choice;
    for(;;)
    {
        printf(" \nIMPLEMENTATION OF CIRCULAR QUEUE\n");
        printf("\nMENU\n");
        printf("\n1.INSERT\n 2.DELETE\n 3.DISPLAY\n 4.EXIT\n\n");
        printf("\nEnter Your Choice::");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
                insert();
                break;
            case 2:
                delete();
                break;
            case 3:
                display();
                break;
            default: exit(0);
        }
        /*end of switch*/
    }
    /*end of for*/
}
/*end of main*/

void insert()
{
    char item;
    if(count==max)
        printf("\nCIRCULAR QUEUE IS FULL\n");
```

```
else
{
    printf("\nEnter the item to be inserted::");
    __fpurge(stdin);
    scanf("%c",&item);
    rear=(rear+1)%max;
    circularq[rear]=item;
    count++;
}/*end of else*/
}/*end of insert function*/
void delete()
{
    char ch;
    if(count==0)
        printf("\nCIRCULAR QUEUE IS EMPTY\n");
    else
    {
        ch=circularq[front];
        printf("\nThe deleted element is::");
        printf("%c",ch);
        front=(front+1)%max;
        count--;
    }/*end of else*/
}/*end of delete function*/
void display()
{
    int i,j;
    char item;
    if(count==0)
        printf("CIRCULAR QUEUE IS EMPTY\n");
    else
    {
        printf("\nContents of the circular queue are:\n");
        j=front;
        for(i=1;i<=count;i++)
        {
            item=circularq[j];
            printf("%c",item);
            j=(j+1)%max;
            printf("\t");
        }/*end of for*/
    }/*end of else*/
}/*end of display function*/
```

OUTPUT:

IMPLEMENTATION OF CIRCULAR QUEUE

MENU

1. INSERT
2. DELETE
3. DISPLAY
4. EXIT

Enter Your Choice::1

Enter the item to be inserted::a

IMPLEMENTATION OF CIRCULAR QUEUE

MENU

1. INSERT
2. DELETE
3. DISPLAY
4. EXIT

Enter Your Choice::1

Enter the item to be inserted::b

IMPLEMENTATION OF CIRCULAR QUEUE

MENU

1. INSERT
2. DELETE
3. DISPLAY
4. EXIT

Enter Your Choice::1

Enter the item to be inserted::c

IMPLEMENTATION OF CIRCULAR QUEUE

MENU

1. INSERT

- 2.DELETE
- 3.DISPLAY
- 4.EXIT

Enter Your Choice::3

Contents of the circular queue are:

a b c

IMPLEMENTATION OF CIRCULAR QUEUE

MENU

- 1.INSERT
- 2.DELETE
- 3.DISPLAY
- 4.EXIT

Enter Your Choice::1

Enter the item to be inserted::d

IMPLEMENTATION OF CIRCULAR QUEUE

MENU

- 1.INSERT
- 2.DELETE
- 3.DISPLAY
- 4.EXIT

Enter Your Choice::1

Enter the item to be inserted::f

IMPLEMENTATION OF CIRCULAR QUEUE

MENU

- 1.INSERT
- 2.DELETE
- 3.DISPLAY
- 4.EXIT

Enter Your Choice::3

Contents of the circular queue are:

a b c d f

IMPLEMENTATION OF CIRCULAR QUEUE

MENU

1. INSERT
2. DELETE
3. DISPLAY
4. EXIT

Enter Your Choice::1

CIRCULAR QUEUE IS FULL

IMPLEMENTATION OF CIRCULAR QUEUE

MENU

1. INSERT
2. DELETE
3. DISPLAY
4. EXIT

Enter Your Choice::2

The deleted element is::a

IMPLEMENTATION OF CIRCULAR QUEUE

MENU

1. INSERT
2. DELETE
3. DISPLAY
4. EXIT

Enter Your Choice::2

The deleted element is::b

IMPLEMENTATION OF CIRCULAR QUEUE

MENU

1. INSERT
2. DELETE
3. DISPLAY
4. EXIT

Enter Your Choice::1

Enter the item to be inserted::x

IMPLEMENTATION OF CIRCULAR QUEUE

MENU

1. INSERT
2. DELETE
3. DISPLAY
4. EXIT

Enter Your Choice::1

Enter the item to be inserted::y

IMPLEMENTATION OF CIRCULAR QUEUE

MENU

1. INSERT
2. DELETE
3. DISPLAY
4. EXIT

Enter Your Choice::1

CIRCULAR QUEUE IS FULL

IMPLEMENTATION OF CIRCULAR QUEUE

MENU

1. INSERT
2. DELETE
3. DISPLAY
4. EXIT

Enter Your Choice::3

Contents of the circular queue are:

c d f x y

IMPLEMENTATION OF CIRCULAR QUEUE

MENU

1. INSERT
2. DELETE
3. DISPLAY
4. EXIT

Enter Your Choice::2

The deleted element is::c

IMPLEMENTATION OF CIRCULAR QUEUE

MENU

1. INSERT
2. DELETE
3. DISPLAY
4. EXIT

Enter Your Choice::2

The deleted element is::d

IMPLEMENTATION OF CIRCULAR QUEUE

MENU

1. INSERT
2. DELETE
3. DISPLAY
4. EXIT

Enter Your Choice::2

The deleted element is::f

IMPLEMENTATION OF CIRCULAR QUEUE

MENU

1. INSERT
2. DELETE
3. DISPLAY
4. EXIT

Enter Your Choice::1

Enter the item to be inserted::z

IMPLEMENTATION OF CIRCULAR QUEUE

MENU

1. INSERT
2. DELETE
3. DISPLAY
4. EXIT

Enter Your Choice::3

Contents of the circular queue are:

x y z

IMPLEMENTATION OF CIRCULAR QUEUE

MENU

1. INSERT
2. DELETE
3. DISPLAY
4. EXIT

Enter Your Choice::4

PROGRAM 7.

7. Design, Develop and Implement a menu driven Program in C for the following operations on Singly Linked List (SLL) of Student Data with the fields: *USN, Name, Branch, 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[12],name[20],branch[10],phno[15];
    int sem;
    struct node *link;
};

typedef struct node* NODE;
NODE start=NULL;
NODE start1=NULL;

NODE create_node()
{
    NODE ptr;
    ptr=(NODE)malloc(sizeof(struct node));
    if(ptr==NULL)
    {
        printf("\nINSUFFICIENT MEMORY\n");
        exit(0);
    }/*end of if*/
    printf("\n\nEnter student data such as USN, NAME,BRANCH,PHNO,SEM\n\n");
    scanf("%s%s%s%s%d",ptr->usn,ptr->name,ptr->branch,ptr->phno,&ptr->sem);
    ptr->link=NULL;
    return ptr;
}/*end of function create_node*/
```

```
NODE insert_front(NODE start)
{
    NODE ptr;
    ptr=create_node();
    if(start== NULL)
        start= ptr;
    else
    {
        ptr->link = start;
        start = ptr;
    }/*end of else*/
    return start;
}/*end of insert_front function*/
NODE insert_end(NODE start)
{
    NODE ptr,temp;
    ptr=create_node();
    if(start==NULL)
        start=ptr;
    else
    {
        temp=start;
        while(temp->link!=NULL)
        {
            temp=temp->link;
        }
        temp->link=ptr;
    }/*end of else*/
    return start;
}/*end of function insert_end*/
NODE delete_front(NODE start)
{
    NODE temp;
    if(start == NULL)
        printf("\n\nLIST EMPTY\n\n");
    else
    {
        temp = start;
        start= start->link;
        printf("\n\nDeleted node is\n");
        printf(" |%s|%s|%s|%s|%d|\n\n",temp->usn,temp->name,temp->branch,temp->phno,
            temp->sem);
        free(temp);
    }/*end of else*/
    return start;
}/*end of function delete_front*/
```

```
NODE delete_end(NODE start)
{
    NODE p,temp;
    if(start==NULL)
    {
        printf("\n\nLIST EMPTY\n\n");
        return;
    }
    temp=start;
    if(temp->link==NULL) /*list with only one node*/
    {
        start=NULL;
    } /*end of if*/
    else /*list with multiple nodes */
    {
        temp=start;
        while(temp->link!=NULL)
        {
            p=temp;
            temp=temp->link;
        } /*end of while*/
        p->link=NULL;
    } /*end of else*/
    printf("\n\nDeleted node is\n\n");
    printf("\n\n |%s|%s|%s|%s|%d|",temp->usn,temp->name,temp->branch, temp->phno,temp->sem);
    free(temp);
    return start;
} /*end of function delete_front*/

void display(NODE start)
{
    NODE temp;
    if(start== NULL)
        printf("\n\nLIST EMPTY\n\n");
    else
    {
        int count=0;
        temp = start;
        printf("\n\nSINGLY LINKED LIST IS: \n\n");
        while(temp != NULL)
        {
            printf("|%s|%s|%s|%s|%d|->",temp->usn,temp->name,temp->branch,
                temp->phno,temp->sem);
            temp=temp->link;
            count++;
        } /*end of while*/
    }
}
```

```
        printf("\n\nNUMBER OF NODE IN LIST::%d\n\n",count);
    }/*end of else*/
}/*end of function display*/
NODE list_stack(NODE ststart1)
{
    int choice;
    printf("\n\nOPERATIONS ON STACK\n\n");
    for(;;)
    {
        printf("\n\nLIST AS STACK\n\n");
        printf("\n1:PUSH\n2:POP\n3:DISPLAY\n4:EXIT\n");
        printf("\nEnter choice::\n");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:start1=insert_front(start1);
                                break;
            case 2:start1=delete_front(start1);
                                break;
            case 3:display(start1);
                                break;
            default:exit(0);
        }
    }/*end of switch*/
}/*end of for*/
}
void main()
{
    int ch,n,i;
    for(;;)
    {
        printf("\nSINGLY LINKED LIST OPERATIONS\n");
        printf("\nMENU\n");
        printf("\n\n1:List creation using Insert front\n2:Display and count\n3:Insert at end\n\n4:Delete at end\n5:List as Stack\n");
        printf("Enter choice::");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:printf("\n\nEnter how many student information you want enter::");
                    scanf("%d",&n);
                    for(i=1;i<=n;i++)
                    {
                        printf("\n\nEnter infomation of student %d\n\n",i);
                        start=insert_front(start);
                    }
                    break;
```

```
        case 2: display(start);
                break;
        case 3: start= insert_end(start);
                break;
        case 4: start=delete_end(start);
                break;
        case 5: start1=list_stack(start1);
                break;
        default: exit(0);
    } /*end of switch*/
} /*end of for*/
} /*end of function main*/
```

OUTPUT:**SINGLY LINKED LIST OPERATIONS****MENU**

1:List creation using Insert front
2:Display and count
3:List as Stack
4:Insert at end
5:Delete at end

Enter choice::1

Enter how many student information you want to enter:3

Enter information of student 1

Enter student data such as USN, NAME,BRANCH ,SEM, PHNO

1ks13cs001
Raj
CSE
1
9288891919

Enter information of student 2

Enter student data such as USN, NAME,BRANCH, SEM, PHNO

1KS13CS002
Sharath
CSE
2

8991791829

Enter information of student 3

Enter student data such as USN, NAME,BRANCH, SEM, PHNO

1KS13CS004

Bharath

CSE

4

7822781426

SINGLY LINKED LIST OPERATIONS

MENU

1:List creation using Insert front

2:Display and count

3:List as Stack

4:Insert at end

5:Delete at end

Enter choice::2

SINGLY LINKED LIST IS:

|1KS13CS004|Bharath|CSE|4|7822781426|->|1KS13CS002|Sharath|CSE|2|8991791829|->|

1ks13cs001|Raj|CSE|1|9288891919|->

NUMBER OF NODE IN LIST::3

SINGLY LINKED LIST OPERATIONS

MENU

1:List creation using Insert front

2:Display and count

3:List as Stack

4:Insert at end

5:Delete at end

Enter choice::4

Enter student data such as USN, NAME,BRANCH,PHNO,SEM

1KS13CS005

Chinnu

CSE
9838712347
4

SINGLY LINKED LIST OPERATIONS

MENU

- 1:List creation using Insert front
- 2:Display and count
- 3:List as Stack
- 4:Insert at end
- 5:Delete at end

Enter choice::2

SINGLY LINKED LIST IS:

|1KS13CS004|Bharath|CSE|7822781426|4|->|1KS13CS002|Sharath|CSE|8991791829|2|->|
1ks13cs001|Raj|CSE|9288891919|1|->|1KS13CS005|Chinnu|CSE|9838712347|4|->

NUMBER OF NODE IN LIST::4

SINGLY LINKED LIST OPERATIONS

MENU

- 1:List creation using Insert front
- 2:Display and count
- 3:List as Stack
- 4:Insert at end
- 5:Delete at end

Enter choice::5

Deteted node is
|1KS13CS005|Chinnu|CSE|9838712347|4|

SINGLY LINKED LIST OPERATIONS

MENU

- 1:List creation using Insert front
- 2:Display and count
- 3:List as Stack
- 4:Insert at end

5:Delete at end

Enter choice::2

SINGLY LINKED LIST IS:

|1KS13CS004|Bharath|CSE|7822781426|4|->|1KS13CS002|Sharath|CSE|8991791829|2|->|1ks13cs001|Raj|CSE|9288891919|1|->

NUMBER OF NODE IN LIST::3

SINGLY LINKED LIST OPERATIONS

MENU

- 1:List creation using Insert front
- 2:Display and count
- 3:List as Stack
- 4:Insert at end
- 5:Delete at end

Enter choice::5

Deteted node is

|1ks13cs001|Raj|CSE|9288891919|1|

SINGLY LINKED LIST OPERATIONS

MENU

- 1:List creation using Insert front
- 2:Display and count
- 3:List as Stack
- 4:Insert at end
- 5:Delete at end

Enter choice::2

SINGLY LINKED LIST IS:

|1KS13CS004|Bharath|CSE|7822781426|4|->|1KS13CS002|Sharath|CSE|8991791829|2|->

NUMBER OF NODE IN LIST::2

SINGLY LINKED LIST OPERATIONS

MENU

- 1:List creation using Insert front
- 2:Display and count
- 3:List as Stack
- 4:Insert at end
- 5:Delete at end

Enter choice::3

OPERATIONS ON STACK

LIST AS STACK

- 1:PUSH
- 2:POP
- 3:DISPLAY
- 4:EXIT

Enter choice::

1

Enter student data such as USN, NAME,BRANCH,PHNO,SEM

1KS13CS006

VINU

CSE

3

8927679912

LIST AS STACK

- 1:PUSH
- 2:POP
- 3:DISPLAY
- 4:EXIT

Enter choice::

1

Enter student data such as USN, NAME,BRANCH,PHNO,SEM

1KS13CS008

DURUTH

CSE

5

8292777826

LIST AS STACK

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

Enter choice::

1

Enter student data such as USN, NAME,BRANCH,PHNO,SEM

IKS13CS010
BHARATH
CSE
8977561328
7

LIST AS STACK

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

Enter choice::

3

SINGLY LINKED LIST IS:

|IKS13CS010|BHARATH|CSE|8977561328|7|->|1KS13CS008|DURUTH|CSE|8292777826|5|->|
1KS13CS006|VINU|CSE|8927679912|3|->

NUMBER OF NODE IN LIST::3

LIST AS STACK

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

Enter choice::

2

Deteted element is |IKS13CS010|BHARATH|CSE|8977561328|7|

LIST AS STACK

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

Enter choice::4

PROGRAM 8.

8. Design, Develop and Implement 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
{
    struct node *llink;
    int ssn;
    char name[20],dept[20],desi[20], phno[12];
    float sal;
    struct node *rlink;
};
typedef struct node* NODE;
NODE start=NULL;
NODE create_node()
{
    NODE ptr;
    ptr=(NODE)malloc(sizeof(struct node));
    if(ptr==NULL)
    {
        printf("\n\nINSUFFICIENT MEMORY\n\n");
        exit(0);
    }
    printf("Enter employee data such as SSN, NAME,DEPARTMENT,DESIGNATION,
        PHONENO AND SALARY\n");
    scanf("%d%s%s%s%s%f",&ptr->ssn,ptr->name,ptr->dept,ptr->desi,ptr->phno,&ptr->sal);
    ptr->llink=NULL;
    ptr->rlink=NULL;
    return ptr;
}/*end of create_node function*/

NODE insert_front(NODE start)
{
    NODE ptr;
    ptr=create_node();
    if(start==NULL)
        start=ptr;
```

```
else
{
    ptr->rlink=start;
    start->llink=ptr;
    start=ptr;
}/*end of else*/
return start;
}/*end of insert_front function*/
NODE insert_end(NODE start)
{
    NODE ptr,temp;
    ptr=create_node();
    if(start==NULL)
        start=ptr;
    else
    {
        temp=start;
        while(temp->rlink!=NULL)
        {
            temp=temp->rlink;
        }
        temp->rlink=ptr;
        ptr->llink=temp;
    }/*end of else*/
    return start;
}/*end of insert_end function*/
NODE delete_front(NODE start)
{
    NODE temp;
    if(start==NULL)
        printf("\n\nDoubly linked List is empty:\n\n");
    else
    {
        temp=start;
        start=start->rlink;
        if(start!=NULL) /*if dll contains more than one node this condition satisfies otherwise
                        if dll contains only one node on moving start it becomes null */
        {
            start->llink=NULL;
        }
        printf("\n\nDELETED FRONT NODE IS::\n\n");
        printf("\n\n%d|%s|%s|%s|%s|\n\n",temp->:ssn,temp->name,temp->dept,
            temp->desi,temp->phno,temp->sal);
        free(temp);
    }/*end of else*/
    return start; }/*end of delete_front function*/
```

```
NODE delete_end(NODE start)
{
    NODE temp,p;
    if(start==NULL)
    {
        printf("\n\nDoubly linked List is empty:\n\n");
        return;
    }
    temp=start;
    if(temp->rlink==NULL) /* only one node in list */
    {
        start=NULL;
    } /*end of if*/
    else /* Multiple nodes in list */
    {
        while(temp->rlink!=NULL)
        {
            temp=temp->rlink;
        }
        p=temp->llink;
        p->rlink=NULL;
    } /* end of else */
    printf("\n\nDELETED END NODE IS:\n\n");
    printf("|%d|%s|%s|%s|%s|%f|",temp->:ssn,temp->name,temp->dept,
        temp->desi,temp->phno,temp->sal);
    free(temp);
    return start;
} /* end of delete_end function*/

void display(NODE start)
{
    NODE temp;
    int count=0;
    if(start==NULL)
    {
        printf("\n\nDoubly linked List is empty:\n\n");
        return;
    }
    temp=start;
    printf("\n\nThe nodes of the doubly linked list are:\n\n");
    while(temp!=NULL)
    {
        printf("<->|%d|%s|%s|%s|%s|%f|",temp->:ssn,temp->name,temp->dept,
            temp->desi,temp->phno,temp->sal);

        temp=temp->rlink;
        count++;
    } /* end of while */
}
```

```
printf("\n\nNumber of node in doubly link list are::%d",count);
}/* end of display function */
void main()
{
int ch,n,i;
for( ;;)
{
    printf("\n\nMENU\n\n");
    printf("\n1:CREATE DLL of N EMPLOYEE USING INSERT END\n2:DISPLAY
        and COUNT\n3:INSERT END\n4:DELETE END\n5:INSERT FRONT\n
        6:DELETE FRONT\n7:DLL AS EQUEUE\n8:EXIT\n\n");
    printf("\nEnter the choice:: ");
    scanf("%d",&ch);
    switch(ch)
    {
        case 1: printf("\n\nEnter how many employee information you want to enter::");
            scanf("%d",&n);
            for(i=1;i<=n;i++)
            {
                printf("\n\nEnter informamtion of employee %d\n",i);
                start=insert_end(start);
            }
            break;
        case 2:display(start);
            break;
        case 3:start=insert_end(start);
            break;
        case 4:start=delete_end(start);
            break;
        case 5:start=insert_front(start);
            break;
        case 6: start= delete_front(start);
            break;
        case 7:printf("\n\nDEQUEUE INSERTION AND DELETION AT
            BOTHENDS\n\n");
            printf("\n\nINSERTION AT FRONT OF DEQUEUE\n\n");
            start=insert_front(start);
            display(start);
            printf("\n\nINSERTION AT END OF DEQUEUE\n\n");
            start=insert_end(start);
            display(start);
            printf("\n\nDELETION AT FRONT OF DEQUEUE\n\n");
            start=delete_front(start);
            display(start);
            printf("\n\nDELETION AT END OF DEQUEUE\n\n");
            start=delete_end(start);
```

```
                display(start);
                break;
            default:exit(0);
        }/* end of switch */
    }/* end of for */
}/* end of main*/
```

OUTPUT:

MENU

1:CREATE DLL of N EMPLOYEE USING INSERT END

2:DISPLAY and COUNT

3:INSERT END

4:DELETE END

5:INSERT FRONT

6:DELETE FRONT

7:DLL AS DEQUEUE

8:EXIT

Enter the choice :: 3

Enter employee data such as SSN,NAME,DEPARTMENT,DESIGNATION,PHONENO AND SALARY

300

UVW

SALES

DIRECTOR

85693214

30000

MENU

1:CREATE DLL of N EMPLOYEE USING INSERT END

2:DISPLAY and COUNT

3:INSERT END

4:DELETE END

5:INSERT FRONT

6:DELETE FRONT

7:DLL AS DEQUEUE

8:EXIT

Enter the choice :: 5

Enter employee data such as SSN,NAME,DEPARTMENT,DESIGNATION,PHONENO AND SALARY

50

DEF

PRODUCTION

MANAGER

85265452

25000

MENU

1:CREATE DLL of N EMPLOYEE USING INSERT END

2:DISPLAY and COUNT

3:INSERT END

4:DELETE END

5:INSERT FRONT

6:DELETE FRONT

7:DLL AS DEQUEUE

8:EXIT

Enter the choice :: 2

The nodes of the doubly linked list are :

<->|50|DEF|PRODUCTION|MANAGER|85265452|25000.000000|

<->|100|ABC|SALES|MANAGER|96586985|25000.000000|

<->|200|XYZ|PRODUCTION|DIRECTOR|74125896|25000.000000|

<->|300|UVW|SALES|DIRECTOR|85693214|30000.000000|

Number of nodes in doubly link list are :: 4

MENU

1:CREATE DLL of N EMPLOYEE USING INSERT END

2:DISPLAY and COUNT

3:INSERT END

4:DELETE END

5:INSERT FRONT

6:DELETE FRONT

7:DLL AS DEQUEUE

8:EXIT

Enter the choice :: 4

DELETED END NODE IS ::

|300|UVW|SALES|DIRECTOR|85693214|30000.000000|

MENU

1:CREATE DLL of N EMPLOYEE USING INSERT END

2:DISPLAY and COUNT

3:INSERT END

4:DELETE END

5:INSERT FRONT

6:DELETE FRONT

7:DLL AS DEQUEUE

8:EXIT

Enter the choice :: 6

DELETED FRONT NODE IS ::

|50|DEF|PRODUCTION|MANAGER|85265452|25000.000000|

MENU

1:CREATE DLL of N EMPLOYEE USING INSERT END

2:DISPLAY and COUNT

3:INSERT END

4:DELETE END

5:INSERT FRONT

6:DELETE FRONT

7:DLL AS DEQUEUE

8:EXIT

Enter the choice :: 2

The nodes of the doubly linked list are :

<->|100|ABC|SALES|MANAGER|96586985|25000.000000|

<->|200|XYZ|PRODUCTION|DIRECTOR|74125896|25000.000000|

Number of nodes in doubly link list are :: 2

MENU

1:CREATE DLL of N EMPLOYEE USING INSERT END

2:DISPLAY and COUNT

3:INSERT END

4:DELETE END

5:INSERT FRONT

6:DELETE FRONT

7:DLL AS DEQUEUE

8:EXIT

Enter the choice :: 7

DEQUEUE INSERTION AND DELETION AT BOTH ENDS

INSERTION AT FRONT OF DEQUEUE

Enter employee data such as SSN,NAME,DEPARTMENT,DESIGNATION,PHONENO AND SALARY

25

KLM

SALES

EMPLOYEE

96396396

20000

The nodes of the doubly linked list are :

<->|25|KLM|SALES|EMPLOYEE|96396396|20000.000000|

<->|100|ABC|SALES|MANAGER|96586985|25000.000000|

<->|200|XYZ|PRODUCTION|DIRECTOR|74125896|25000.000000|

Number of nodes in doubly link list are :: 3

INSERTION AT END OF DEQUEUE

Enter employee data such as SSN,NAME,DEPARTMENT,DESIGNATION,PHONENO AND SALARY

250

STR

PRODUCTION

EMPLOYEE

78945602

20000

The nodes of the doubly linked list are :

<->|25|KLM|SALES|EMPLOYEE|96396396|20000.000000|

<->|100|ABC|SALES|MANAGER|96586985|25000.000000|

<->|200|XYZ|PRODUCTION|DIRECTOR|74125896|25000.000000|

<->|250|STR|PRODUCTION|EMPLOYEE|78945602|20000.000000|

Number of nodes in doubly link list are :: 4

DELETION AT FRONT OF DEQUEUE

DELETED FRONT NODE IS ::

|25|KLM|SALES|EMPLOYEE|96396396|20000.000000|

The nodes of the doubly linked list are :

<->|100|ABC|SALES|MANAGER|96586985|25000.000000|

<->|200|XYZ|PRODUCTION|DIRECTOR|74125896|25000.000000|

<->|250|STR|PRODUCTION|EMPLOYEE|78945602|20000.000000|

Number of nodes in doubly link list are :: 3

DELETION AT END OF DEQUEUE

DELETED END NODE IS ::

|250|STR|PRODUCTION|EMPLOYEE|78945602|20000.000000|

The nodes of the doubly linked list are :

<->|100|ABC|SALES|MANAGER|96586985|25000.000000|

<->|200|XYZ|PRODUCTION|DIRECTOR|74125896|25000.000000|

Number of nodes in doubly link list are :: 2

MENU

1:CREATE DLL of N EMPLOYEE USING INSERT END

2:DISPLAY and COUNT

3:INSERT END

4:DELETE END

5:INSERT FRONT

6:DELETE FRONT

7:DLL AS DEQUEUE

8:EXIT

Enter the choice :: 8

9. Design, Develop and Implement 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>
struct node
{
    int cf,px,py,pz,flag;
    struct node *link;
};
typedef struct node *NODE;

NODE create_node()
{
    NODE ptr;
    ptr=(NODE) malloc (sizeof(struct node));
    if(ptr==NULL)
    {
        printf("Insufficient memory\n");
        exit(0);
    }
    return ptr;
}/*end of create_node*/

NODE insert_end(int cf,int x,int y,int z,NODE head)
{
    NODE temp,ptr;
    ptr=create_node();
    ptr->cf=cf;
    ptr->px=x;
    ptr->py=y;
    ptr->pz=z;
    ptr->flag=0;
    if(head->link==head)
    {
        head->link=ptr;
        ptr->link=head;
    }
}
```

```
else
{
    temp=head->link;
    while(temp->link!=head)
        temp=temp->link;
    temp->link=ptr;
    ptr->link=head;
}/*end of insert_end*/
return head;
}

void display(NODE head)
{
    NODE temp;
    if(head->link==head)
        printf("\n\nPolynomial empty\n");
    else
    {
        temp=head->link;
        while(temp!=head)
        {
            if(temp->cf<0)
                printf("%dx^%dy^%dz^%d",temp->cf,temp->px,temp->py,temp->pz);
            else
                printf("+%dx^%dy^%dz^%d",temp->cf,temp->px,temp->py,temp->pz);
            temp=temp->link;
        }/*end of while*/
    }
}/*end of display function*/

NODE add_poly(NODE h1,NODE h2, NODE h3)
{
    NODE p1,p2;
    int cof;
    p1=h1->link;
    while(p1!=h1)
    {
        p2=h2->link;
        while(p2!=h2)
        {
            if(p1->px==p2->px && p1->py==p2->py && p1->pz==p2->pz)
                break;
            p2=p2->link;
        }/*end of inner while*/
    }
}
```

```
    if(p2!=h2)
    {
        cof=p1->cf+p2->cf;
        p2->flag=1;
        if(cof!=0)
            h3=insert_end(cof,p1->px,p1->py,p1->pz,h3);
    }/*end of if*/
    else
        h3=insert_end(p1->cf,p1->px,p1->py,p1->pz,h3);
    p1=p1->link;
}/*end of outer while*/
p2=h2->link;
while(p2!=h2)
{
    if(p2->flag==0)
        h3=insert_end(p2->cf,p2->px,p2->py,p2->pz,h3);
    p2=p2->link;
}/*end of while*/
return h3;
}/*end of add_poly*/

NODE read_poly(NODE head)
{
    int i,cf,x,y,z;
    printf("\n\nEnter the coeffecient and exponents to stop polynomial reading enter -999\n\n");
    for(i=1;;i++)
    {
        printf("\nEnter the %d term\n",i);
        printf("coeff=");
        scanf("%d",&cf);
        if(cf==-999)
            break;
        printf("pow x=");
        scanf("%d",&x);
        printf("pow y=");
        scanf("%d",&y);
        printf("pow z=");
        scanf("%d",&z);
        head=insert_end(cf,x,y,z,head);
    }/*end of for*/
    return head;
}/*end of read_poly*/
```

```
void polysum()
{
    NODE h1,h2,h3;
    h1=create_node();
    h2=create_node();
    h3=create_node();
    h1->link=h1;
    h2->link=h2;
    h3->link=h3;
    printf("\n\nEnter the first polynominal\n\n");
    h1=read_poly(h1);
    printf("\n\nEnter the second polynominal\n\n");
    h2=read_poly(h2);
    h3=add_poly(h1,h2,h3);
    printf("\n\nThe first polynominal is\n\n");
    display(h1);
    printf("\n\nSecond polynominal is\n\n");
    display(h2);
    printf("\n\nThe sum of two polynominal is\n\n");
    display(h3);
}/*end of polysum*/

void eval()
{
    NODE h,temp;
    int x,y,z,sum=0;
    h=create_node();
    h->link=h;
    printf("\n\nEnter the polynominal\n\n");
    h=read_poly(h);
    printf("\n\nPolynominal is\n\n");
    display(h);
    printf("\n\nEnter the values of variables x,y and z\n\n");
    scanf("%d%d%d",&x,&y,&z);
    temp=h->link;
    while(temp!=h)
    {
        sum+=temp->cf*pow(x,temp->px)*pow(y,temp->py)*pow(z,temp->pz);
        temp=temp->link;
    }/*end of while*/
    printf("\n\nThe total sum is:: %d\n",sum);
}/*end of eval function*/
```

```
void main()
{
int ch;
for(;;)
{
    printf("\n\n1.Represent and Evaluate\n2.Add Two poly\n3.Exit");
    printf("\n\nEnter Your Choice:");
    scanf("%d",&ch);
    switch(ch)
    {
        case 1: eval(); break;
        case 2: polysum(); break;
        default: exit(0);
    }/*end of switch*/
}/*end of for*/
}/*end of main*/
```

OUTPUT:

1.Represent and Evaluate
2.Add Two poly
3.Exit

Enter Your Choice : 1

Enter the polynomial

Enter the coeffecient and stop polynomial reading enter -999

Enter the 1 term

coeff=6

pow x=2

pow y=2

pow z=1

Enter the 2 term

coeff=-4

pow x=0

pow y=1

pow z=5

Enter the 3 term

coeff=3

pow x=3

pow y=1

pow z=1

Enter the 4 term

```
coeff=2
pow x=1
pow y=5
pow z=1
Enter the 5 term
coeff=-2
pow x=1
pow y=1
pow z=3
Enter the 6 term
coeff=-999
```

Polynomial is
 $+6x^2y^2z^1-4x^0y^1z^5+3x^3y^1z^1+2x^1y^5z^1-2x^1y^1z^3$

Enter the values of variables x,y and z
2 2 3

The total sum is:: -1344

- 1.Represent and Evaluate
- 2.Add Two poly
- 3.Exit

Enter Your Choice : 2

Enter the first polynomial

Enter the coefficient and stop polynomial reading enter -999

```
Enter the 1 term
coeff=4
pow x=2
pow y=3
pow z=3
Enter the 2 term
coeff=-6
pow x=2
pow y=2
pow z=2
Enter the 3 term
coeff=4
pow x=1
pow y=1
pow z=1
Enter the 4 term
coeff=10
```

pow x=0
pow y=0
pow z=0
Enter the 5 term
coeff=-999

Enter the second polynomial

Enter the coefficient and stop polynomial reading enter -999

Enter the 1 term

coeff=5

pow x=0

pow y=3

pow z=5

Enter the 2 term

coeff=4

pow x=2

pow y=2

pow z=2

Enter the 3 term

coeff=8

pow x=1

pow y=1

pow z=1

Enter the 4 term

coeff=-999

The first polynomial is

$+4x^2y^3z^3-6x^2y^2z^2+4x^1y^1z^1+10x^0y^0z^0$

Second polynomial is

$+5x^0y^3z^5+4x^2y^2z^2+8x^1y^1z^1$

The sum of two polynomial is

$+4x^2y^3z^3-2x^2y^2z^2+12x^1y^1z^1+10x^0y^0z^0+5x^0y^3z^5$

1.Represent and Evaluate

2.Add Two poly

3.Exit

Enter Your Choice : 3

PROGRAM 10

Design, Develop and Implement 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

e. Exit

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

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

```
struct node
```

```
{
```

```
    struct node *lchild;
```

```
    int data;
```

```
    struct node *rchild;
```

```
};
```

```
typedef struct node* NODE;
```

```
NODE root=NULL;
```

```
NODE create_node()
```

```
{
```

```
NODE ptr;
```

```
ptr=(NODE)malloc(sizeof(struct node));
```

```
if(ptr==NULL)
```

```
{
```

```
    printf("Insufficient memory\n");
```

```
    exit(0);
```

```
}
```

```
else
```

```
{
```

```
    printf("Enter data: ");
```

```
    scanf("%d",&ptr->data);
```

```
    ptr->lchild=ptr->rchild=NULL;
```

```
}/* end of else*/
```

```
return ptr;
```

```
}/* end of create_node function*/
```

```
NODE create_bst(NODE root)
```

```
{
```

```
    NODE ptr, temp, p;
```

```
ptr=create_node();
```

```
if(root==NULL)
```

```
{
```

```
    root=ptr;
```

```
    return root;
```

```
}/* end of if*/
```

```
p=NULL;
temp=root;
while(temp!=NULL)
{
    p=temp;
    if(ptr->data==temp->data)
    {
        printf("Duplicate items are not allowed\n\n");
        free(ptr);
        return root;
    }/* end of if*/
    if(ptr->data<temp->data)
        temp=temp->lchild;
    else
        temp=temp->rchild;
}/* end of while*/
if(ptr->data<p->data)
    p->lchild=ptr;
else
    p->rchild=ptr;
return root;
}/* end of create_bst*/

void inorder(NODE t)
{
    if(t!=NULL)
    {
        inorder(t->lchild);
        printf("%d\t",t->data);
        inorder(t->rchild);
    }/* end of if*/
}/* end of inorder*/

void preorder(NODE t)
{
    if(t!=NULL)
    {
        printf("%d\t",t->data);
        preorder(t->lchild);
        preorder(t->rchild);
    }/* end of if*/
}/* end of preorder*/
```

```
void postorder(NODE t)
{
if(t!=NULL)
{
    postorder(t->lchild);
    postorder(t->rchild);
    printf("%d\t",t->data);
}/* end of if*/
}/* end of postorder*/

void traverse(NODE root)
{
if(root==NULL)
    printf("Empty tree\n\n");
else
{
    printf("\n\nPreorder traversal:\t");
    preorder(root);
    printf("\n\nInorder traversal:\t");
    inorder(root);
    printf("\n\nPostorder traversal:\t");
    postorder(root);
}/* end of else*/
}/* end of traversal function*/

void search_bst(NODE root)
{
    int key,flag=0;
    NODE temp;
    printf("Enter key to search in bst::");
    scanf("%d",&key);
    if(root==NULL)
    {
        printf("BST IS EMPTY\n\n ");
        return;
    }
    else
    {
        temp=root;
        while(temp!=NULL)
        {
            if(key==temp->data)
            {
                printf("Key found\n\n");
                flag=1;
                break;
            }/* end of if*/
        }
    }
}
```

```
        else if(key<temp->data)
            temp=temp->lchild;
        else
            temp=temp->rchild;
    }/* end of while*/
}/* end of else*/
if(flag==0)
    printf("Key not found\n\n");
}/* end of search function*/

void main()
{
    int ch;
    for(;;)
    {
        printf("\n\nOPERATIONS ON BST\n\n");
        printf("\n\n1:CREATE BST \n2:TRAVERSE BST\n3:SEARCH\n4:EXIT\n\n");
        printf("Enter your choice::");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:root=create_bst(root);
                    break;
            case 2: traverse(root);
                    break;
            case 3:search_bst(root);
                    break;
            default:exit(0);
        }/* end of switch*/
    }/* end of for*/
}/* end of main*/
```

OUTPUT:

OPERATIONS ON BST

```
1:CREATE BST
2:TRAVERSE BST
3:SEARCH
4:EXIT
Enter your choice::1
Enter data:: 6
```

OPERATIONS ON BST

1:CREATE BST
2:TRAVERSE BST
3:SEARCH
4:EXIT
Enter your choice::1
Enter data:: 9

OPERATIONS ON BST

1:CREATE BST
2:TRAVERSE BST
3:SEARCH
4:EXIT
Enter your choice::1
Enter data:: 5

OPERATIONS ON BST

1:CREATE BST
2:TRAVERSE BST
3:SEARCH
4:EXIT
Enter your choice::1
Enter data:: 2

OPERATIONS ON BST

1:CREATE BST
2:TRAVERSE BST
3:SEARCH
4:EXIT
Enter your choice::1
Enter data:: 8

OPERATIONS ON BST

1:CREATE BST
2:TRAVERSE BST
3:SEARCH
4:EXIT
Enter your choice::1
Enter data:: 15

OPERATIONS ON BST

1:CREATE BST
2:TRAVERSE BST

3:SEARCH

4:EXIT

Enter your choice::1

Enter data:: 24

OPERATIONS ON BST

1:CREATE BST

2:TRAVERSE BST

3:SEARCH

4:EXIT

Enter your choice::1

Enter data:: 14

OPERATIONS ON BST

1:CREATE BST

2:TRAVERSE BST

3:SEARCH

4:EXIT

Enter your choice::1

Enter data:: 7

OPERATIONS ON BST

1:CREATE BST

2:TRAVERSE BST

3:SEARCH

4:EXIT

Enter your choice::1

Enter data:: 8

Duplicate items are not allowed

OPERATIONS ON BST

1:CREATE BST

2:TRAVERSE BST

3:SEARCH

4:EXIT

Enter your choice::1

Enter data:: 5

Duplicate items are not allowed

OPERATIONS ON BST

1:CREATE BST

2:TRAVERSE BST

3:SEARCH

4:EXIT

Enter your choice::1

Enter data:: 2

Duplicate items are not allowed

OPERATIONS ON BST

1:CREATE BST

2:TRAVERSE BST

3:SEARCH

4:EXIT

Enter your choice::2

Preorder traversal: 6 5 2 9 8 7 1514 24

Inorder traversal: 2 5 6 7 8 9 1415 24

Postorder traversal: 2 5 7 8 14 24 159 6

OPERATIONS ON BST

1:CREATE BST

2:TRAVERSE BST

3:SEARCH

4:EXIT

Enter your choice::3

Enter key to search in bst::24

Key found

OPERATIONS ON BST

1:CREATE BST

2:TRAVERSE BST

3:SEARCH

4:EXIT

Enter your choice::3

Enter key to search in bst::10

Key not found

OPERATIONS ON BST

1:CREATE BST

2:TRAVERSE BST

3:SEARCH

4:EXIT

Enter your choice::4

PROGRAM 11

11. Design, Develop and Implement 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**

11 a) BFS PROGRAM

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

void bfs(int a[10][10],int n,int u)
{
    int front,rear,q[10],visited[10],v,i;
    for(i=0;i<n;i++)
        visited[i]=0;
    front=0;
    rear=-1;
    printf("\nThe nodes visited from %d are::\n",u);
    q[++rear]=u;
    visited[u]=1;
    printf("%d\t",u);
    while(front<=rear)
    {
        u=q[front++];
        for(v=0;v<n;v++)
        {
            if(a[u][v]==1&&visited[v]==0)
            {
                printf("%d\t",v);
                visited[v]=1;
                q[++rear]=v;
            }/* end of if*/
        }/* end of for*/
    }/* end of while*/
}/* end of bfs function*/

void main()
{
    int a[10][10],n,source,i,j;
    printf("\nEnter number of vertices:");
    scanf("%d",&n);
    printf("\nEnter the graph as adjacency matrix\n");
```

```

for(i=0;i<n;i++)
{
    for(j=0;j<n;j++)
        scanf("%d",&a[i][j]);
}
printf("Enter source vertex::");
scanf("%d",&source);
printf("\nBFS OF GIVEN GRAPH\n");
bfs(a,n,source);
}

```

OUTPUT:

1)

Enter number of vertices::4

Enter the graph as adjacency matrix

0 1 1 0

0 0 1 1

0 0 0 1

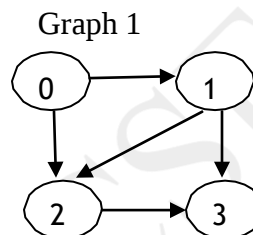
0 0 0 0

Enter source vertex::0

BFS OF GIVEN GRAPH

The nodes visited from 0 are::

0 1 2 3



2)

Enter number of vertices::5

Enter the graph as adjacency matrixGraph 2

0 1 1 0 0

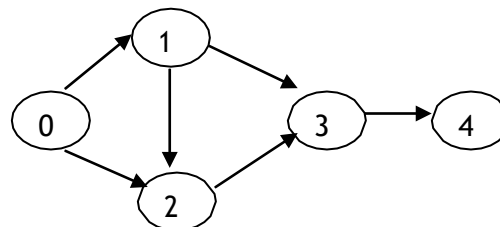
0 0 1 1 0

0 0 0 1 0

0 0 0 0 1

0 0 0 0 0

Enter source vertex::1



BFS OF GIVEN GRAPH

The nodes visited from 1 are::

1 2 3 4

11 b) DFS PROGRAM

```
#include<stdio.h>
int a[10][10],visited[10],n;

void dfs(int u)
{
    int v;
    visited[u]=1;
    printf("%d\t",u);
    for(v=0;v<n;v++)
    {
        if(a[u][v]==1&&visited[v]==0)
            dfs(v); /* end of if*/
    }/* end of dfs*/

void main()
{
    int source,i,j;
    printf("Enter number of vertices::");
    scanf("%d",&n);
    printf("\n\nEnter graph as adjacency matrix form\n\n");
    for(i=0;i<n;i++)
    {
        for(j=0;j<n;j++)
            scanf("%d",&a[i][j]);
    }/* end of for*/
    for(i=0;i<n;i++)
        visited[i]=0;
    printf("\n\nEnter source vertex::");
    scanf("%d",&source);
    printf("\nDFS of given graph\n");
    printf("Node visited from source node %d are::\n",source);
    dfs(source);
}
```

OUTPUT:

1)

Enter number of vertices::4

Enter graph as adjacency matrix form

0 1 1 0

0 0 1 1

0 0 0 1

(for Graph 1)

0 0 0 0

Enter source vertex::0

DFS of given graph

Node visited from source node 0 are::

0 1 2 3

2)

Enter number of vertices::5

Enter graph as adjacency matrix form

0 1 1 0 0

0 0 1 1 1

0 0 0 1 0 (for Graph 2)

0 0 0 0 1

0 0 0 0 0

Enter source vertex::0

DFS of given graph

Node visited from source node 0 are::

0 1 2 3 4

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. Design and develop a Program in C that uses Hash function $H: K \rightarrow L$ as $H(K) = K \bmod m$ (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>
#include<string.h>
struct record
{
    int empno,flag;// employee number is the key- K with 4 digits
    char name[10];
}emp[100];
int hash(int m)
{
    int r;
    r=m%100;
    return r;
}
void main()
{
    int m,k,eno,loc,i,n,j;
    char name[10];
    FILE *in;
    printf("\nEnter no of records to read from file: ");
    scanf("%d",&n);
    in = fopen("input.txt", "r");
    if(n<=10)
    {
        for(k=0;k<100;k++)
            emp[k].flag=0;
        for(i=0;i<n;i++)
        {
            fscanf(in,"%d%s",&eno,name);
            loc=hash(eno);
            if(emp[loc].flag==0)
            {
                printf("\nRecord:%d is mapped to address:%d\n",i,loc);
                emp[loc].empno=eno;
                emp[loc].flag=1;
                strcpy(emp[loc].name,name);
            }
        }
    }
    /* end of if*/
}
```

```

else
{
    printf("\n\nCollision occured for record %d resolved using
        linear probing\n\n",i);
    for(j=loc+1;j<100;j++)
    {
        if(emp[j].flag==0)
        {
            printf("\nRecord:%d is at address:%d\n",i,j);
            strcpy(emp[j].name,name);
            emp[j].empno=eno;
            emp[j].flag=1;
            break;
        }/* end of if*/
    }/* end of for*/
    if(j>=100)
    {
        printf("HASH TABLE IS FULL\n");
        printf("\n----- \n");
    }/* end of if*/
    }/* end of else*/
}/* end of for*/
fclose(in);
printf("\n\nThe Hash Table Content is: ");
for(i=0;i<100;i++)
{
    if(emp[i].flag==1)
        printf("\n%d\t%d\t%s",i,emp[i].empno,emp[i].name);
    else
        printf("\n####");
}/* end of for*/
}
else
    printf("\nFile is containing only 10 records\n\n");
}/* end of main*/

```

OUTPUT:

```

Enter no of records to read from file: 10
Record:0 is mapped to address:1
Record:1 is mapped to address:10
Collision occured for record 2 resolved using linear probing
Record:2 is at address:11
Collision occured for record 3 resolved using linear probing
Record:3 is at address:12
Record:4 is mapped to address:36

```

9999Collision occured for record 5 resolved using linear probing

Record:5 is at address:37

Record:6 is mapped to address:56

Collision occured for record 7 resolved using linear probing

Record:7 is at address:57

Record:8 is mapped to address:66

Record:9 is mapped to address:89

The Hash Table Content is:

Address Empno Name

#####

1 1201 aaa

#####

#####

#####

#####

#####

#####

#####

#####

10 2410 bbb

11 3510 ccc

12 4610 ddd

#####

#####

#####

#####

#####

#####

#####

#####

#####

#####

#####

#####

#####

#####

#####

#####

#####

#####

#####

#####

#####

#####

```
####  
####  
36  7836  Eee  
37  8936  Fff  
####  
####  
####  
####  
####  
####  
####  
####  
####  
####  
####  
####  
####  
####  
####  
####  
####  
####  
####  
####  
56  1356  Gg  
      g  
57  3456  Hh  
      h  
####  
####  
####  
####  
####  
####  
####  
####  
####  
66  1466  Iii  
####  
####  
####  
####  
####  
####  
####  
####  
####  
####  
####  
####  
####  
####  
####  
####
```

```
####  
####  
####  
####  
####  
####  
####  
####  
####  
89 4589 jjj ####  
####  
####  
####  
####  
####  
####  
####  
####  
####
```

DEPT OF CSE KSTIT

Possible Viva questions

1. What are some of the applications for the tree data structure?
 - 1- Manipulation of the arithmetic expressions.
 - 2- Symbol table construction.
 - 3- Syntax analysis.
2. What is the data structures used to perform recursion?

Stack. Because of its LIFO (Last In First Out) property it remembers its 'caller' so knows whom to return when the function has to return. Recursion makes use of system stack for storing the return addresses of the function calls.
3. Convert the expression $((A + B) * C - (D - E) ^ (F + G))$ to equivalent Prefix and Postfix notations.
 1. **Prefix Notation:** $^ - * + ABC - DE + FG$
 2. **Postfix Notation:** $AB + C * DE - - FG + ^$
4. In tree construction which is the suitable efficient data structure?

Linked list is the suitable efficient data structure.
5. What are the types of Collision Resolution Techniques and the methods used in each of the type?
 1. **Open addressing (closed hashing)**, The methods used include: Overflow block.
 2. **Closed addressing (open hashing)**, The methods used include: Linked list, Binary tree.
6. **List out the areas in which data structures are applied extensively?**
 1. Compiler Design,
 2. Operating System,
 3. Database Management System,
 4. Statistical analysis package,
 5. Numerical Analysis,
 6. Graphics,
 7. Artificial Intelligence,
 8. Simulation
7. What is a linked list?

A linked list is a linear collection of data elements, called nodes, where the linear order is given by pointers. Each node has two parts first part contain the information of the element second part contains the address of the next node in the list.
8. What are the advantages of linked list over array (static data structure)?

The disadvantages of array are unlike linked list it is expensive to insert and delete elements in the array One can't double or triple the size of array as it occupies block of memory space. In linked list each element in list contains a field, called a link or pointer which contains the address of the next element Successive element's need not occupy adjacent space in memory.
9. Can we apply binary search algorithm to a sorted linked list, why?

No we cannot apply binary search algorithm to a sorted linked list, since there is no way of indexing the middle element in the list. This is the drawback in using linked list as a data structure
10. What do you mean by overflow and underflow?

When new data is to be inserted into the data structure but there is no available space i.e. free storage list is empty this situation is called overflow.

When we want to delete data from a data structure that is empty this situation is called underflow.

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

12. What is a priority queue? The priority queue is a data structure in which the intrinsic ordering of the elements (numeric or alphabetic) determines the result of its basic operation. It is of two types i) Ascending priority queue- Here smallest item can be removed (insertion is arbitrary) ii) Descending priority queue- Here largest item can be removed (insertion is arbitrary)

13. What are the disadvantages of sequential storage?

.Fixed amount of storage remains allocated to the data structure even if it contains less element. No more than fixed amount of storage is allocated causing overflow

14. Define circular list?

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

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

15. What are the disadvantages of circular list?

i) We can't traverse the list backward ii) If a pointer to a node is given we cannot delete the node

16. Define double linked list?

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

- i) An info field that contains the information stored in the node
- ii) Left field that contains pointer to node on left side
- iii) Right field that contains pointer to node on right side

16. For an undirected graph with n vertices and e edges, the sum of the degree of each vertex is equal to Ans: $2e$

17. A full binary tree with $2n+1$ nodes contains

Ans: n non-leaf nodes

18. If a node in a BST has two children, then its inorder predecessor has

Ans: no right child

19. A binary tree in which if all its levels except possibly the last, have the maximum number of nodes and all the nodes at the last level appear as far left as possible, is known as

Ans: complete binary tree

20. A linear list of elements in which deletion can be done from one end (front) and insertion can take place only at the other end (rear) is known as a

Ans: queue

21. In a circular linked list

Ans: there is no beginning and no end.

22. The data structure required for Breadth First Traversal on a graph is

Ans: queue

23. The data structure required to evaluate a postfix expression is (A) queue (B) stack (C) array (D) linked-list
Ans: Stack
24. The data structure required to check whether an expression contains balanced parenthesis is
Ans: Stack
25. The number of leaf nodes in a complete binary tree of depth d is
Ans: 2^d
26. A linear collection of data elements where the linear node is given by means of pointer is called (Ans: linked list
27. Representation of data structure in memory is known as:
Ans: abstract data type
28. If the address of $A[1][1]$ and $A[2][1]$ are 1000 and 1010 respectively and each element occupies 2 bytes then the array has been stored in _____ order.
Ans: row major
29. An adjacency matrix representation of a graph cannot contain information of :
Ans: parallel edges