

ACS COLLEGE OF ENGINEERING

(Accredited by NBA, NAAC Approved by A.I.C.T.E. New Delhi, Recognized by Govt. of Karnataka & Affiliated to V.T.U.,
Kambipura, Mysore Road, Bangalore-560074)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



LABORATORY MANUAL

DATA STRUCTURES LABORATORY

Academic Year 2024-25



Subject Code: BCSL305
(For III Semester BE)

Prepared by

Department of CSE-Cyber Security

ACS College of Engineering
Kambipura, Mysore Road, Bengaluru-560074

LIST OF EXPERIMENTS**DATA STRUCTURES LABORATORY
SEMESTER-III**

CourseCode	BCSL305	CIE Marks	50
Number of Contact Hours/Week	0:0:2	SEE Marks	50
Total Number of Lab Contact Hours	28	Exam Hours	03

Credits- 1**Course Learning Objectives:**

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

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

Descriptions (if any):

- Implement all the programs in “C ” Programming Language and Linux OS.

Programs List:

1.	<p>Develop a Program in C for the following:</p> <p>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).</p> <p>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.</p>
2.	<p>Develop a Program in C for the following operations on Strings.</p> <p>a. Read a main String (STR), a Pattern String (PAT) and a Replace String (REP)</p> <p>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</p> <p>Support the program with functions for each of the above operations. Don't use Built-in functions.</p>
3.	<p>Develop a menu driven Program in C for the following operations on STACK of Integers (Array Implementation of Stack with maximum size MAX)</p> <p>a. Push an Element on to Stack</p> <p>d. Demonstrate Overflow and Underflow situations on Stack</p> <p>Support the program with appropriate functions for each of the above operations</p>
4.	<p>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.</p>
5.	<p>Develop a Program in C for the following Stack Applications</p> <p>a. Evaluation of Suffix expression with single digit operands and operators: +, -, *, /, %, ^</p> <p>b. Solving Tower of Hanoi problem with n disks</p>

6.	<p>Develop a menu driven Program in C for the following operations on Circular QUEUE of Characters (Array Implementation of Queue with maximum size MAX)</p> <ol style="list-style-type: none"> 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 <p>Support the program with appropriate functions for each of the above operations</p>
7.	<p>Develop 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></p>
8.	<p>Develop 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></p> <ol style="list-style-type: none"> 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
9.	<p>Develop a Program in C for the following operations on Singly Circular Linked List (SCLL) with header nodes</p> <ol style="list-style-type: none"> 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) <p>Support the program with appropriate functions for each of the above operations</p>
10.	<p>Develop a menu driven Program in C for the following operations on Binary Search Tree (BST) of Integers .</p> <ol style="list-style-type: none"> 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.	<p>Develop a Program in C for the following operations on Graph(G) of Cities</p> <ol style="list-style-type: none"> 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	<p>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: $K \rightarrow L$ 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.</p>

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 their applications
- Use appropriate searching and sorting algorithms for the give 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)
 - 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. 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.**

ALGORITHM:

1. Define a structure Day with fields: char* dayName, int date, and char* activity.
2. Create create() function to allocate memory and read day name, date, and activity.
3. Create read() function to loop through the calendar array and call create() for each day.
4. Create display() function to print all details stored in the calendar.
5. In main(), read the number of days and allocate memory for calendar array using malloc.
6. Call read() to input details for each day.
7. Call display() to show all the week's activity details.
8. Call freeMemory() to deallocate memory for strings in each structure.
9. Free the calendar array itself.
10. End the program.

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h> // Structure to represent a day in the calendar
struct Day
{
char * dayName; // Dynamically allocated string for the day name
int date;
char * activity; // Dynamically allocated string for the activity description
}; // Function to create a day
void create(struct Day * day) {
// Allocate memory for the day name and activity
day -> dayName = (char * ) malloc(sizeof(char) * 20); // Assuming day names
are less than 20 characters
day -> activity = (char * ) malloc(sizeof(char) * 100); // Assuming activity
```

```
descriptions are less than 100 characters// Input the day details
printf("Enter the day name:");
scanf("%s", day -> dayName);
printf("Enter the date:");
scanf("%d", & day -> date);
printf("Enter the activity for the day:");
scanf(" %", day -> activity); // Read the entire line, including spaces
} // Function to read data from the keyboard and create the calendar
void read(struct Day * calendar, int size) {
for (int i = 0; i < size; i++) {
printf("Enter details for Day %d:\n", i + 1);
create( & calendar[i]);
}
} // Function to display the calendar
void display(struct Day * calendar, int size) {
printf("\nWeek's Activity Details:\n");
for (int i = 0; i < size; i++) {
printf("Day %d:\n", i + 1);
printf("Day Name: %s\n", calendar[i].dayName);
printf("Date: %d\n", calendar[i].date);
printf("Activity: %s\n", calendar[i].activity);
printf("\n");
}
} // Function to free the dynamically allocated memory
void freeMemory(struct Day * calendar, int size) {
for (int i = 0; i < size; i++) {
free(calendar[i].dayName);
free(calendar[i].activity);
}
}
int main() {
int size;
printf("Enter the number of days in the week:");
scanf("%d", & size); // Dynamically allocate memory for the calendar
struct Day * calendar = (struct Day * ) malloc(sizeof(struct Day) * size); //
Check if memory allocation is successful
if (calendar == NULL) {
printf("Memory allocation failed. Exiting program.\n");
return 1;
} // Read and display the calendar
read(calendar, size);
```

```
display(calendar, size);// Free the dynamically allocated memory
freeMemory(calendar, size);// Free the memory allocated for the calendar array
free(calendar);
return 0;
}
```

SAMPLE OUTPUT:

Enter the number of days in the week: 7
Enter details for Day 1:
Enter the day name: Sunday
Enter the date: 1
Enter the activity for the day: Learning
Enter details for Day 2:
Enter the day name: Monday
Enter the date: 2
Enter the activity for the day: Coding
Enter details for Day 3:
Enter the day name: Tuesday
Enter the date: 3
Enter the activity for the day: Testing
Enter details for Day 4:
Enter the day name: Wednesday
Enter the date: 4
Enter the activity for the day: Debugging
Enter details for Day 5:
Enter the day name: Thursday
Enter the date: 5
Enter the activity for the day: Publishing
Enter details for Day 6:
Enter the day name: Friday
Enter the date: 6
Enter the activity for the day: Marketing
Enter details for Day 7:
Enter the day name: Saturday
Enter the date: 7
Enter the activity for the day: Earning
Week's Activity Details:
Day 1:
Day Name: Sunday

Date: 1

Activity: Learning

Day 2:

Day Name: Monday

Date: 2

Activity: CodingDay 3:

Day Name: Tuesday

Date: 3

Activity: TestingDay 4:

Day Name: Wednesday

Date: 4

Activity: DebuggingDay 5:

Day Name: Thursday

Date: 5

Activity: PublishingDay 6:

Day Name: Friday

Date: 6

Activity: MarketingDay 7:

Day Name: Saturday

Date: 7

Activity: Earning

PROGRAM-2

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.

ALGORITHM

1. Declare global character arrays `str`, `pat`, `rep`, `res` and integer variables `c`, `m`, `i`, `j`, `k`, `flag`.
2. Read the main string `str`, pattern string `pat`, and replacement string `rep` from the user.
3. Initialize pointers `c`, `m`, `i`, `j`, `flag` to 0.
4. Traverse the main string `str` using `c` until `\0` (end of string) is reached.
5. Compare characters of `str` with `pat` starting at positions `m` and `i` respectively.
6. If characters match, increment `i` and `m`, and check if `pat` is fully matched.
7. If full match is found, set `flag=1` and copy `rep` into `res` at current position `j`.
8. Reset `i=0`, and set `c=m` to continue from the next character.
9. If characters do not match, copy `str[c]` to `res[j]`, then increment `c` and `j`, and reset `m=c`, `i=0`.
10. Terminate `res` with `\0` after loop ends.
11. If `flag==1`, display `res` as the updated string; else display "Pattern not found".

PROGRAM

```
#include<stdio.h>
char str[50], pat[20], rep[20], res[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++) {
res[j] = rep[k];
}
i = 0;
c = m;
} else {
```

```
res[j] = str[c];
j++;
c++;
m = c;
i = 0;
}
}
res[j] = '\0';
}
void main() {
printf("Enter the main string:");
gets(str);
printf("\nEnter the pat string:");
gets(pat);
printf("\nEnter the replace string:");
gets(rep);
printf("\nThe string before pattern match is:\n %s", str);
stringmatch();
if (flag == 1)
printf("\nThe string after pattern match and replace is: \n %s ", res);
else
printf("\nPattern string is not found");
}
```

Sample output:

Enter the main string:Designed by vtucode

Enter the pat string:vtucode

Enter the replace string:Braham

The string before pattern match is:Designed by vtucode

The string after pattern match and replace is:Designed by Braham

PROGRAM-3

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

Algorithm:

1. **Step 1:** START
2. **Step 2:** Define MAX as the maximum stack size (e.g., 3).
3. **Step 3:** Initialize `stack[MAX]` and `top = -1`.
4. **Step 4:** Repeat the following steps in a loop until user selects Exit:
5. **Step 4.1:** Display menu:
 - a. Push
 - b. Pop
 - c. Check Palindrome
 - d. Display Stack
 - e. Exit
6. **Step 4.2:** Read user choice.
7. **Step 4.3:** If choice is 1 (Push):
If `top == MAX - 1`, display "Stack Overflow".
Else increment `top` and read value into `stack[top]`.
8. **Step 4.4:** If choice is 2 (Pop):
If `top == -1`, display "Stack Underflow".
Else print `stack[top]` and decrement `top`.
9. **Step 4.5:** If choice is 3 (Palindrome Check):
Display stack from top to bottom and bottom to top.
If `stack[i] == stack[top - i]` for all `i` up to `top/2`, print "Palindrome".
Else print "Not a palindrome".
10. **Step 4.6:** If choice is 4 (Display):
If `top == -1`, display "Stack is empty".
Else print all elements from `stack[top]` to `stack[0]`.
11. **Step 4.7:** If choice is 5 (Exit):
Terminate the loop.
12. **Step 5:** STOP

Program:

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 3 //you can change this size according to your requirement
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");
}
}
```

SAMPLE 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.ExitEnter your choice: 1

Enter an element to be pushed: 11~~~~~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.ExitEnter your choice: 1

Enter an element to be pushed: 12~~~~~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.ExitEnter your choice: 1

Enter an element to be pushed: 13~~~~~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.ExitEnter your choice: 1

Enter an element to be pushed: 14

~~~~Stack overflow~~~~~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.ExitEnter your choice: 4

Stack elements are:

| 13 |

| 12 |

| 11 |~~~~~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.ExitEnter your choice: 2

Element popped is: 13~~~~~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.ExitEnter your choice: 4

Stack elements are:

| 12 |

| 11 |~~~~~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.ExitEnter your choice: 2

Element popped is: 12~~~~~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.ExitEnter your choice: 2

Element popped is: 11~~~~~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.ExitEnter your choice: 2

~~~~Stack underflow~~~~~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.ExitEnter your choice: 4

~~~~Stack is empty~~~~~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.ExitEnter your choice: 1

Enter an element to be pushed: 11~~~~~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.ExitEnter your choice: 1

Enter an element to be pushed: 22~~~~~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.ExitEnter your choice: 1

Enter an element to be pushed: 11~~~~~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.ExitEnter your choice: 3

Stack content are:

| 11 |

| 22 |

| 11 |Reverse of stack content are:

| 11 |

| 22 |

| 11 |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.ExitEnter your choice: 2

Element popped is: 11~~~~~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.ExitEnter your choice: 2

Element popped is: 22~~~~~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.ExitEnter your choice: 1

Enter an element to be pushed: 33~~~~~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.ExitEnter your choice: 1

Enter an element to be pushed: 22~~~~~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.ExitEnter your choice: 3

Stack content are:

| 22 |

| 33 |

| 11 |Reverse of stack content are:

| 11 |

| 33 |

| 22 |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.ExitEnter your choice: 5

PROGRAM-4

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.

Algorithm:

1. **Step 1:** START
2. **Step 2:** Read the infix expression as a string.
3. **Step 3:** Initialize an empty stack and push a sentinel symbol #.
4. **Step 4:** Repeat for each symbol in the infix expression:
 5. **Step 4.1:** If the symbol is an operand (letter or digit), append it to the postfix expression.
 6. **Step 4.2:** If the symbol is (, push it onto the stack.
 7. **Step 4.3:** If the symbol is), pop and append from the stack until (is found; discard both parentheses.
 8. **Step 4.4:** If the symbol is an operator:
 - a. While the precedence of top of stack is **greater than or equal** to current operator:
 - i. Pop and append to postfix.
 - b. Push the current operator onto the stack.
9. **Step 5:** After scanning the entire infix expression, pop all remaining operators from the stack and append to postfix.
10. **Step 6:** Terminate postfix expression with null character \0.
11. **Step 7:** Display the original infix expression and the converted postfix expression.
12. **Step 8:** STOP

Program:

```
#include<stdio.h>
#include<stdlib.h>
void evaluate();
void push(char);
char pop();
int prec(char);
char infix[30], postfix[30], stack[30];
int top = -1;void main() {
printf("\nEnter the valid infix expression:");
scanf("%s", infix);
evaluate();
printf("\nThe entered infix expression is :\n %s \n", infix);
printf("\nThe corresponding postfix expression is :\n %s \n", postfix);
}void evaluate() {
int i = 0, j = 0;
char symb, temp;push('#');for (i = 0; infix[i] != '\0'; i++)
```

```
{
symb = infix[i];
switch (symb) {
case '(':
push(symb);
break;case ')':
temp = pop();
while (temp != '(') {
postfix[j] = temp;
j++;
temp = pop();
}
break;
case '+':
case '-':
case '*':
case '/':
case '%':
case '^':
case '$':
while (prec(stack[top]) >= prec(symb)) {
temp = pop();
postfix[j] = temp;
j++;
}
push(symb);
break;
default:
postfix[j] = symb;
j++;
}
}
while (top > 0) {
temp = pop();
postfix[j] = temp;
j++;
}
postfix[j] = '\0';
}void push(char item) {
top = top + 1;
```

```
stack[top] = item;
}char pop() {
char item;
item = stack[top];
top = top - 1;
return item;
}int prec(char symb) {
int p;
switch (symb) { case '#':
p = -1;
break;case '(':
case ')':
p = 0;
break;case '+':
case '-':
p = 1;
break;case '*':
case '/':
case '%':
p = 2;
break;case '^':
case '$':
p = 3;
break;
}
return p;
}
```

Sample output:

Enter the valid infix expression:(a+b)*c/d^5%1

The entered infix expression is :

(a+b)*c/d^5%1The corresponding postfix expression is :

ab+c*d5^/1%

PROGRAM-5**5. Develop a Program in C for the following Stack Applications****a. Evaluation of Suffix expression with single digit operands and operators:****+, -, *, /, %, ^****Algorithm:**

1. **Step 1:** START
2. **Step 2:** Read the postfix expression as a string.
3. **Step 3:** Initialize an empty stack and set `top = -1`.
4. **Step 4:** Repeat for each character in the postfix expression:
 5. **Step 4.1:** If the character is a digit, convert it to integer and push it onto the stack.
 6. **Step 4.2:** If the character is an operator:
 - a. Pop two elements from the stack: first `op2`, then `op1`.
 - b. Perform the operation `op1 operator op2`.
 - c. Push the result back onto the stack.
7. **Step 5:** After scanning all characters, pop the final result from the stack.
8. **Step 6:** Display the result.
9. **Step 7:** STOP

Program:

```
#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);
}
```

Sample output :

Enter a valid postfix expression:

623+-382/+*2\$3+

Result = 52

PROGRAM-5(b)**5b. Solving Tower of Hanoi problem with n disks****Algorithm:**

1. **Step 1:** START
2. **Step 2:** Read the number of disks n .
3. **Step 3:** Call the recursive function `tower(n, source, temp, destination)`.
4. **Step 4:** In `tower(n, source, temp, destination)`:
5. **Step 4.1:** If $n == 0$, return (base case).
6. **Step 4.2:** Recursively move $n-1$ disks from `source` to `temp` using `destination` as auxiliary.
7. **Step 4.3:** Print move of disk n from `source` to `destination`.
8. **Step 4.4:** Recursively move $n-1$ disks from `temp` to `destination` using `source` as auxiliary.
9. **Step 5:** After all moves, calculate and print total number of moves as $2^n - 1$.
10. **Step 6:** STOP

Program:

```
#include <stdio.h>

void tower(int n, int source, int temp, int destination)
{
    if (n == 0)
        return;
    tower(n - 1, source, destination, temp);
    printf("\nMove disc %d from %c to %c", n, source, destination);
    tower(n - 1, temp, source, destination);
}

void main() {
    int n;
    printf("\nEnter the number of discs: \n");
    scanf("%d", &n);
    tower(n, 'A', 'B', 'C');
    printf("\n\nTotal Number of moves are: %d", (int) pow(2, n) - 1);
}
```

Sample Output:

```
Enter the number of discs: 3
Move disc 1 from A to C
Move disc 2 from A to B
Move disc 1 from C to B
Move disc 3 from A to C
Move disc 1 from B to A
Move disc 2 from B to C
```

Move disc 1 from A to C

Total Number of moves are: 7

PROGRAM-6

6. Develop a menu driven Program in C for the following operations on Circular QUEUE of

Characters (Array Implementation of Queue with maximum size MAX)

a. Insert an Element on to Circular QUEUE

b. Delete an Element from Circular QUEUE

c. Demonstrate Overflow and Underflow situations on Circular QUEUE

d. Display the status of Circular QUEUE

e. Exit

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

Algorithm:

- a. **Start**
- b. Initialize `front = -1, rear = -1`, and define an array `cq[MAX]`
- c. Repeat the following steps until user chooses Exit:
- d. Display the menu with 4 choices
- e. Read user's choice
- f. If choice is 1 (Insert):
- g. Check if `(front == (rear + 1) % MAX)`, if true, display overflow
- h. Else, if `front == -1`, set `front = rear = 0`, else set `rear = (rear + 1) % MAX`
- i. Insert the item at `cq[rear]`
- j. If choice is 2 (Delete):
- k. If `front == -1`, display underflow
- l. Else, print `cq[front]` as deleted element
- m. If `front == rear`, set `front = rear = -1`, else set `front = (front + 1) % MAX`
- n. If choice is 3 (Display):
- o. If `front == -1`, display empty queue message
- p. Else, print all elements from `front` to `rear` in circular fashion
- q. If choice is 4, exit the program
- r. Else, display invalid choice message
- s. **Stop**

Program:

```
#include <stdio.h>
#include<stdlib.h>
#include<stdio_ext.h>
#define MAX 3char cq[MAX];
int front = -1, rear = -1;void insert(char);
void delete();
void display();
void main() {
int ch;
char item;
```

```
while (1) {
printf("\n\n~~Main Menu~~");
printf("\n==> 1. Insertion and Overflow Demo");
printf("\n==> 2. Deletion and Underflow Demo");
printf("\n==> 3. Display");
printf("\n==> 4. Exit");
printf("\nEnter Your Choice: ");
scanf("%d", & ch);
__fpurge(stdin);
switch (ch) {
case 1:
printf("\n\nEnter the element to be inserted: ");
scanf("%c", & item);
insert(item);
break;
case 2:
delete();
break;
case 3:
display();
break;
case 4:
exit(0);
default:
printf("\n\nPlease enter a valid choice");
}
}
}void insert(char item) {
if (front == (rear + 1) % MAX) {
printf("\n\n~~Circular Queue Overflow~~");
} else {
if (front == -1)
front = rear = 0;
else
rear = (rear + 1) % MAX;
cq[rear] = item;
}
}void delete() {
char item;
```

```
if (front == -1) {
printf("\n\n~~Circular Queue Underflow~~");
} else {
item = cq[front];
printf("\n\nDeleted element from the queue is: %c ", item);if (front == rear)
//only one element
front = rear = -1;
else
front = (front + 1) % MAX;
}
}void display() {
int i;
if (front == -1) {
printf("\n\nCircular Queue Empty");
} else {
printf("\n\nCircular Queue contents are:\n");
printf("Front[%d]-> ", front);
for (i = front; i != rear; i = (i + 1) % MAX) {
printf(" %c", cq[i]);
}
printf(" %c", cq[i]);
printf(" <-[%d]Rear", rear);
}
}
```

Sample Output:

~~Main Menu~~

==> 1. Insertion and Overflow Demo

==> 2. Deletion and Underflow Demo

==> 3. Display

==> 4. ExitEnter Your Choice: 1

Enter the element to be inserted: A~~Main Menu~~

==> 1. Insertion and Overflow Demo

==> 2. Deletion and Underflow Demo

==> 3. Display

==> 4. ExitEnter Your Choice: 1

Enter the element to be inserted: B~~Main Menu~~

- ==> 1. Insertion and Overflow Demo
- ==> 2. Deletion and Underflow Demo
- ==> 3. Display
- ==> 4. ExitEnter Your Choice: 1

Enter the element to be inserted: C~~Main Menu~~

- ==> 1. Insertion and Overflow Demo
- ==> 2. Deletion and Underflow Demo
- ==> 3. Display
- ==> 4. ExitEnter Your Choice: 1

Enter the element to be inserted: D

~~Circular Queue Overflow~~~~Main Menu~~

- ==> 1. Insertion and Overflow Demo
- ==> 2. Deletion and Underflow Demo
- ==> 3. Display
- ==> 4. ExitEnter Your Choice: 3

Circular Queue contents are:

Front[0]-> A B C <-[2]Rear~~Main Menu~~

- ==> 1. Insertion and Overflow Demo
- ==> 2. Deletion and Underflow Demo
- ==> 3. Display
- ==> 4. ExitEnter Your Choice: 2

Deleted element from the queue is: A~~Main Menu~~

- ==> 1. Insertion and Overflow Demo
- ==> 2. Deletion and Underflow Demo
- ==> 3. Display
- ==> 4. ExitEnter Your Choice: 3

Circular Queue contents are:

Front[1]-> B C <-[2]Rear~~Main Menu~~

- ==> 1. Insertion and Overflow Demo
- ==> 2. Deletion and Underflow Demo
- ==> 3. Display
- ==> 4. ExitEnter Your Choice: 4

PROGRAM-7

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**

Algorithm

- a. **Start**
- b. Initialize `start = NULL` and `count = 0`
- c. Display the main menu with 6 choices: Create, Display, Insert End, Delete End, Stack Demo, Exit
- d. Read user's main menu choice
- e. If choice is 1:
- f. Read number of students `n`
- g. Repeat `n` times: call `insertfront()` and update `start`
- h. If choice is 2: call `display()` to show SLL and number of nodes
- i. If choice is 3: call `insertend()` and update `start`
- j. If choice is 4: call `deleteend()` and update `start`
- k. If choice is 5:
- l. Display stack sub-menu with choices: Push, Pop, Display, Exit
- m. Repeat until stack demo exit is selected:
- n. If push: call `insertfront()`
- o. If pop: call `deletefront()`
- p. If display: call `display()`
- q. If choice is 6: Exit the program
- r. If invalid choice: Display error and return to main menu
- s. **Stop**

Program:

```
#include<stdio.h>#include<stdlib.h>struct node {
char usn[25], name[25], branch[25];
int sem;
long int phone;
struct node * link;
};
typedef struct node * NODE;NODE start = NULL;
```

```
int count = 0;NODE create() {
NODE snode;
snode = (NODE) malloc(sizeof(struct node));if (snode == NULL) {
printf("\nMemory is not available");
exit(1);
}
printf("\nEnter the usn,Name,Branch, sem,PhoneNo of the student:");
scanf("%s %s %s %d %ld", snode -> usn, snode -> name, snode -> branch, &
snode -> sem, & snode -> phone);
snode -> link = NULL;
count++;
return snode;
}NODE insertfront() {
NODE temp;
temp = create();
if (start == NULL) {
return temp;
}temp -> link = start;
return temp;
}NODE deletefront() {
NODE temp;
if (start == NULL) {
printf("\nLinked list is empty");
return NULL;
}if (start -> link == NULL) {
printf("\nThe Student node with usn:%s is deleted ", start -> usn);
count--;
free(start);
return NULL;
}
temp = start;
start = start -> link;
printf("\nThe Student node with usn:%s is deleted", temp -> usn);
count--;
free(temp);
return start;
}NODE insertend() {
NODE cur, temp;
temp = create();if (start == NULL) {
return temp;
```

```
}
cur = start;
while (cur -> link != NULL) {
cur = cur -> link;
}
cur -> link = temp;
return start;
}NODE deleteend() {
NODE cur, prev;
if (start == NULL) {
printf("\nLinked List is empty");
return NULL;
}if (start -> link == NULL) {
printf("\nThe student node with the usn:%s is deleted", start -> usn);
free(start);
count--;
return NULL;
}prev = NULL;
cur = start;
while (cur -> link != NULL) {
prev = cur;
cur = cur -> link;
}printf("\nThe student node with the usn:%s is deleted", cur -> usn);
free(cur);
prev -> link = NULL;
count--;
return start;
}void display() {
NODE cur;
int num = 1;if (start == NULL) {printf("\nNo Contents to display in SLL \n");
return;
}
printf("\nThe contents of SLL: \n");
cur = start;
while (cur != NULL) {
printf("\n||%d|| USN:%s| Name:%s| Branch:%s| Sem:%d| Ph:%ld|", num, cur ->
usn, cur -> name, cur -> branch, cur -> sem, cur -> phone);
cur = cur -> link;
num++;
}
```

```
printf("\n No of student nodes is %d \n", count);
}void stackdemo() {
int ch;
while (1) {
printf("\n~~~Stack Demo using SLL~~~\n");
printf("\n1:Push operation \n2: Pop operation \n3: Display \n4:Exit \n");
printf("\nEnter your choice for stack demo:");
scanf("%d", & ch);switch (ch) {
case 1:
start = insertfront();
break;
case 2:
start = deletefront();
break;
case 3:
display();
break;
default:
return;
}
}
return;
}int main() {
int ch, i, n;
while (1) {
printf("\n~~~Menu~~~");
printf("\nEnter your choice for SLL operation \n");
printf("\n1:Create SLL of Student Nodes");
printf("\n2:DisplayStatus");
printf("\n3:InsertAtEnd");
printf("\n4:DeleteAtEnd");
printf("\n5:Stack Demo using SLL(Insertion and Deletion at Front)");
printf("\n6:Exit \n");
printf("\nEnter your choice:");
scanf("%d", & ch);switch (ch) {
case 1:
printf("\nEnter the no of students: ");
scanf("%d", & n);
for (i = 1; i <= n; i++)
start = insertfront();
```

```
break;case 2:
display();
break;case 3:
start = insertend();
break;case 4:
start = deleteend();
break;case 5:
stackdemo();
break;case 6:
exit(0);default:
printf("\nPlease enter the valid choice");}
}
```

Sample Output:

~~~Menu~~~

Enter your choice for SLL operation1:Create SLL of Student Nodes

2:DisplayStatus

3:InsertAtEnd

4>DeleteAtEnd

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

6:ExitEnter your choice:1Enter the no of students: 3

Enter the usn,Name,Branch, sem,PhoneNo of the student:

1ME21CS017

Braham

CSE

5

8768586443Enter the usn,Name,Branch, sem,PhoneNo of the student:

1ME21CS015

Bikash

CSE

5

8734687996Enter the usn,Name,Branch, sem,PhoneNo of the student:

1ME21AI015

Shoaib

AI&ML

5

6748353877~~~Menu~~~

Enter your choice for SLL operation1:Create SLL of Student Nodes

2:DisplayStatus

3:InsertAtEnd

4:DeleteAtEnd

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

6:ExitEnter your choice:2

The contents of SLL:||1|| USN:1ME21AI015| Name:Shoaib| Branch:AI&ML| Sem:5| Ph:6748353877|

||2|| USN:1ME21CS015| Name:Bikash| Branch:CSE| Sem:5| Ph:8734687996|

||3|| USN:1ME21CS017| Name:Braham| Branch:CSE| Sem:5| Ph:8768586443|

No of student nodes is 3~~~Menu~~~

Enter your choice for SLL operation1:Create SLL of Student Nodes

2:DisplayStatus

3:InsertAtEnd

4:DeleteAtEnd

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

6:ExitEnter your choice:3Enter the usn,Name,Branch, sem,PhoneNo of the student:

1ME21CS068

Rajan

CSE

5

3426527765~~~Menu~~~

Enter your choice for SLL operation1:Create SLL of Student Nodes

2:DisplayStatus

3:InsertAtEnd

4:DeleteAtEnd

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

6:ExitEnter your choice:2

The contents of SLL:||1|| USN:1ME21AI015| Name:Shoaib| Branch:AI&ML| Sem:5| Ph:6748353877|

||2|| USN:1ME21CS015| Name:Bikash| Branch:CSE| Sem:5| Ph:8734687996|

||3|| USN:1ME21CS017| Name:Braham| Branch:CSE| Sem:5| Ph:8768586443|

||4|| USN:1ME21CS068| Name:Rajan| Branch:CSE| Sem:5| Ph:3426527765|

No of student nodes is 4~~~Menu~~~

Enter your choice for SLL operation1:Create SLL of Student Nodes

2:DisplayStatus

3:InsertAtEnd

4:DeleteAtEnd

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

6:ExitEnter your choice:4

The student node with the usn:1ME21CS068 is deleted~~~Menu~~~

Enter your choice for SLL operation1:Create SLL of Student Nodes

2:DisplayStatus

3:InsertAtEnd

4:DeleteAtEnd

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

6:ExitEnter your choice:2

The contents of SLL:||1|| USN:1ME21AI015| Name:Shoaib| Branch:AI&ML|  
Sem:5| Ph:6748353877|

||2|| USN:1ME21CS015| Name:Bikash| Branch:CSE| Sem:5| Ph:8734687996|

||3|| USN:1ME21CS017| Name:Braham| Branch:CSE| Sem:5| Ph:8768586443|

No of student nodes is 3~~~Menu~~~

Enter your choice for SLL operation1:Create SLL of Student Nodes

2:DisplayStatus

3:InsertAtEnd

4:DeleteAtEnd

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

6:ExitEnter your choice:4

The student node with the usn:1ME21CS017 is deleted~~~Menu~~~

Enter your choice for SLL operation1:Create SLL of Student Nodes

2:DisplayStatus

3:InsertAtEnd

4:DeleteAtEnd

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

6:ExitEnter your choice:5

~~~Stack Demo using SLL~~~1:Push operation

2: Pop operation

3: Display

4:ExitEnter your choice for stack demo:1Enter the usn,Name,Branch,
sem,PhoneNo of the student:

1ME21CS005

Aman

CSE

5

6587594335~~~Stack Demo using SLL~~~1:Push operation

2: Pop operation

3: Display

4:ExitEnter your choice for stack demo:3

The contents of SLL:|1| USN:1ME21CS005| Name:Aman| Branch:CSE|
Sem:5| Ph:6587594335|
|2| USN:1ME21AI015| Name:Shoaib| Branch:AI&ML| Sem:5| Ph:6748353877|
|3| USN:1ME21CS015| Name:Bikash| Branch:CSE| Sem:5| Ph:8734687996|
No of student nodes is 3~~~Stack Demo using SLL~~~1: Push operation
2: Pop operation
3: Display
4: ExitEnter your choice for stack demo:1Enter the usn,Name,Branch,
sem,PhoneNo of the student:
1ME21CS092
Shubham
CSE
5
9869754354~~~Stack Demo using SLL~~~
1:Push operation
2: Pop operation
3: Display
4:ExitEnter your choice for stack demo:3
The contents of SLL:|1| USN:1ME21CS092| Name:Shubham| Branch:CSE|
Sem:5| Ph:9869754354|
|2| USN:1ME21CS005| Name:Aman| Branch:CSE| Sem:5| Ph:6587594335|
|3| USN:1ME21AI015| Name:Shoaib| Branch:AI&ML| Sem:5| Ph:6748353877|
|4| USN:1ME21CS015| Name:Bikash| Branch:CSE| Sem:5| Ph:8734687996|
No of student nodes is 4~~~Stack Demo using SLL~~~1:Push operation
2: Pop operation
3: Display
4:ExitEnter your choice for stack demo:2
The Student node with usn:1ME21CS092 is deleted~~~Stack Demo using
SLL~~~1:Push operation
2: Pop operation
3: Display
4:ExitEnter your choice for stack demo:3
The contents of SLL:|1| USN:1ME21CS005| Name:Aman| Branch:CSE|
Sem:5| Ph:6587594335|
|2| USN:1ME21AI015| Name:Shoaib| Branch:AI&ML| Sem:5| Ph:6748353877|
|3| USN:1ME21CS015| Name:Bikash| Branch:CSE| Sem:5| Ph:8734687996|
No of student nodes is 3~~~Stack Demo using SLL~~~1: Push operation

2: Pop operation

3: Display

4: ExitEnter your choice for stack demo:4~~~Menu~~~

Enter your choice for SLL operation1:Create SLL of Student Nodes

2:DisplayStatus

3:InsertAtEnd

4:DeleteAtEnd

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

6:Exit

Enter your choice:6

PROGRAM-8

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**

Algorithm:

- a. **Start**
- b. Initialize `first = NULL, count = 0`
- c. Display main menu with 8 options
- d. Read user's menu choice
- e. If choice is 1:
- f. Read number of employees `n`
- g. Repeat `n` times: call `insertend()` and update `first`
- h. If choice is 2: call `display()` to show all nodes and count
- i. If choice is 3: call `insertend()` and update `first`
- j. If choice is 4: call `deleteend()` and update `first`
- k. If choice is 5: call `insertfront()` and update `first`
- l. If choice is 6: call `deletefront()` and update `first`
- m. If choice is 7:
- n. Display deque menu with 6 choices
- o. Repeat until deque exit is selected:
- p. If 1: call `insertfront()`
- q. If 2: call `deletefront()`
- r. If 3: call `insertend()`
- s. If 4: call `deleteend()`
- t. If 5: call `display()`
- u. If 6: exit deque demo
- v. If choice is 8: terminate program
- w. If invalid choice: display error and repeat menu
- x. **Stop**

Program:

```
#include<stdio.h>
#include<stdlib.h>
struct node {
char ssn[25], name[25], dept[10], designation[25];
```

```
int sal;
long int phone;
struct node * llink;
struct node * rlink;
};
typedef struct node * NODE; NODE first = NULL;
int count = 0; NODE create() {
NODE enode;
enode = (NODE) malloc(sizeof(struct node));
if (enode == NULL) {
printf("\nRunning out of memory");
exit(0);
}
printf("\nEnter the ssn,Name,Department,Designation,Salary,PhoneNo of
the employee: \n");
scanf("%s %s %s %s %d %ld", enode -> ssn, enode -> name, enode ->
dept, enode -> designation, & enode -> sal, & enode -> phone);
enode -> llink = NULL;
enode -> rlink = NULL;

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

```
cur = cur -> rlink;
nodeno++;
}
printf("\nNo of employee nodes is %d", count);
}NODE deletefront() {
NODE temp;
if (first == NULL) {
printf("\nDoubly Linked List is empty");
return NULL;
}
if (first -> rlink == NULL) {
printf("\nThe employee node with the ssn:%s is deleted", first -> ssn);
free(first);
count--;
return NULL;
}
temp = first;
first = first -> rlink;
temp -> rlink = NULL;
first -> llink = NULL;
printf("\nThe employee node with the ssn:%s is deleted", temp -> ssn);
free(temp);
count--;
return first;
}NODE insertend() {
NODE cur, temp;
temp = create();if (first == NULL) {
return temp;
}
cur = first;
while (cur -> rlink != NULL) {
cur = cur -> rlink;
}cur -> rlink = temp;
temp -> llink = cur;
return first;
}NODE deleteend() {
NODE prev, cur;
if (first == NULL) {
printf("\nDoubly Linked List is empty");
return NULL;
```

```
}if (first -> rlink == NULL) {  
printf("\n\nThe employee node with the ssn:%s is deleted", first -> ssn);  
free(first);  
count--;  
return NULL;  
}prev = NULL;  
cur = first;while (cur -> rlink != NULL) {  
prev = cur;  
cur = cur -> rlink;  
}cur -> llink = NULL;  
printf("\n\nThe employee node with the ssn:%s is deleted", cur -> ssn);  
free(cur);  
prev -> rlink = NULL;  
count--;  
return first;  
}void deqdemo() {  
int ch;  
while (1) {  
printf("\n\nDemo Double Ended Queue Operation");  
printf("\n\n1:InsertQueueFront\n 2: DeleteQueueFront\n  
3:InsertQueueRear\n 4:DeleteQueueRear\n 5:DisplayStatus\n 6: Exit \n");  
scanf("%d", & ch);switch (ch) {  
case 1:  
first = insertfront();  
break;  
case 2:  
first = deletefront();  
break;  
case 3:  
first = insertend();  
break;  
case 4:  
first = deleteend();  
break;  
case 5:  
display();  
break;  
default:  
return;  
}  
}
```

```
}  
}void main() {  
int ch, i, n;  
while (1) {  
printf("\n\n~~~Menu~~~");  
printf("\n1:Create DLL of Employee Nodes");  
printf("\n2:DisplayStatus");  
printf("\n3:InsertAtEnd");  
printf("\n4>DeleteAtEnd");  
printf("\n5:InsertAtFront");  
printf("\n6>DeleteAtFront");  
printf("\n7:Double Ended Queue Demo using DLL");  
printf("\n8:Exit \n");  
printf("\nPlease enter your choice: ");  
scanf("%d", & ch);switch (ch) {  
case 1:  
printf("\nEnter the no of Employees: ");  
scanf("%d", & n);  
for (i = 1; i <= n; i++)  
first = insertend();  
break;case 2:  
display();  
break;case 3:  
first = insertend();  
break;case 4:  
first = deleteend();  
break;case 5:  
first = insertfront();  
break;case 6:  
first = deletefront();  
break;case 7:  
deqdemo();  
break;case 8:  
exit(0);  
default:  
printf("\nPlease Enter the valid choice");  
}  
}  
}
```

Sample Output:

~~~Menu~~~

**1:Create DLL of Employee Nodes**

**2:DisplayStatus**

**3:InsertAtEnd**

**4:DeleteAtEnd**

**5:InsertAtFront**

**6:DeleteAtFront**

**7:Double Ended Queue Demo using DLL**

**8:Exit**

**Please enter your choice: 1**

**Enter the no of Employees: 2Enter the**

**ssn,Name,Department,Designation,Salary,PhoneNo of the employee:**

**1EPL**

**Braham**

**Developer**

**Senior**

**13627**

**8476283712Enter the ssn,Name,Department,Designation,Salary,PhoneNo  
of the employee:**

**2EPL**

**Aman**

**Trader**

**Manager**

**20000**

**2763578156~~~Menu~~~**

**1:Create DLL of Employee Nodes**

**2:DisplayStatus**

**3:InsertAtEnd**

**4:DeleteAtEnd**

**5:InsertAtFront**

**6:DeleteAtFront**

**7:Double Ended Queue Demo using DLL**

**8:Exit**

**Please enter your choice:**

**2ENode:1||SSN:1EPL|Name:Braham|Department:Developer|Designation:Senior|Salary:13627|Phone no:8476283712**

**ENode:2||SSN:2EPL|Name:Aman|Department:Trader|Designation:Manager|Salary:20000|Phone no:2763578156**

**No of employee nodes is 2~~~Menu~~~**

**1:Create DLL of Employee Nodes**

**2:DisplayStatus**

**3:InsertAtEnd**

**4>DeleteAtEnd**

**5:InsertAtFront**

**6>DeleteAtFront**

**7:Double Ended Queue Demo using DLL**

**8:Exit**

**Please enter your choice: 3Enter the ssn,Name,Department,Designation,Salary,PhoneNo of the employee:**

**3EPL**

**Bikash**

**Meeting**

**Manager**

**30000**

**8237462936~~~Menu~~~**

**1:Create DLL of Employee Nodes**

**2:DisplayStatus**

**3:InsertAtEnd**

**4>DeleteAtEnd**

**5:InsertAtFront**

**6>DeleteAtFront**

**7:Double Ended Queue Demo using DLL**

**8:Exit**

**Please enter your choice:**

**2ENode:1||SSN:1EPL|Name:Braham|Department:Developer|Designation:Senior|Salary:13627|Phone no:8476283712**

**ENode:2||SSN:2EPL|Name:Aman|Department:Trader|Designation:Manager|Salary:20000|Phone no:2763578156**

**ENode:3||SSN:3EPL|Name:Bikash|Department:Meeting|Designation:Manager|Salary:30000|Phone no:8237462936**

**No of employee nodes is 3~~~Menu~~~**

**1:Create DLL of Employee Nodes**

**2:DisplayStatus**

**3:InsertAtEnd**

**4>DeleteAtEnd**

**5:InsertAtFront**

**6:DeleteAtFront**

**7:Double Ended Queue Demo using DLL**

**8:Exit**

**Please enter your choice: 5Enter the**

**ssn,Name,Department,Designation,Salary,PhoneNo of the employee:**

**4EPL**

**Shoaib**

**Digital Marketing**

**Manager**

**40000**

**2835826437~~~Menu~~~**

**1:Create DLL of Employee Nodes**

**2:DisplayStatus**

**3:InsertAtEnd**

**4:DeleteAtEnd**

**5:InsertAtFront**

**6:DeleteAtFront**

**7:Double Ended Queue Demo using DLL**

**8:ExitPlease enter your choice:**

**2ENode:1||SSN:4EPL|Name:Shoaib|Department:Digital**

**Marketing|Designation:Manager|Salary:40000|Phone no:2835826437**

**ENode:2||SSN:1EPL|Name:Braham|Department:Developer|Designation:Se  
nior|Salary:13627|Phone no:8476283712**

**ENode:3||SSN:2EPL|Name:Aman|Department:Trader|Designation:Manag  
er|Salary:20000|Phone no:2763578156**

**ENode:4||SSN:3EPL|Name:Bikash|Department:Meeting|Designation:Mana  
ger|Salary:30000|Phone no:8237462936**

**No of employee nodes is 4~~~Menu~~~**

**1:Create DLL of Employee Nodes**

**2:DisplayStatus**

**3:InsertAtEnd**

**4:DeleteAtEnd**

**5:InsertAtFront**

**6:DeleteAtFront**

**7:Double Ended Queue Demo using DLL**

**8:Exit**

**Please enter your choice: 4The employee node with the ssn:3EPL is  
deleted~~~Menu~~~**

**1:Create DLL of Employee Nodes**

**2:DisplayStatus**

**3:InsertAtEnd**

**4:DeleteAtEnd**

**5:InsertAtFront**

**6:DeleteAtFront**

**7:Double Ended Queue Demo using DLL**

**8:Exit**

**Please enter your choice: 6The employee node with the ssn:4EPL is deleted~~~Menu~~~**

**1:Create DLL of Employee Nodes**

**2:DisplayStatus**

**3:InsertAtEnd**

**4:DeleteAtEnd**

**5:InsertAtFront**

**6:DeleteAtFront**

**7:Double Ended Queue Demo using DLL**

**8:Exit**

**Please enter your choice:**

**2ENode:1||SSN:1EPL|Name:Braham|Department:Developer|Designation:Senior|Salary:13627|Phone no:8476283712**

**ENode:2||SSN:2EPL|Name:Aman|Department:Trader|Designation:Manager|Salary:20000|Phone no:2763578156**

**No of employee nodes is 2~~~Menu~~~**

**1:Create DLL of Employee Nodes**

**2:DisplayStatus**

**3:InsertAtEnd**

**4:DeleteAtEnd**

**5:InsertAtFront**

**6:DeleteAtFront**

**7:Double Ended Queue Demo using DLL**

**8:Exit**

**Please enter your choice: 7Demo Double Ended Queue Operation**

**1:InsertQueueFront**

**2: DeleteQueueFront**

**3:InsertQueueRear**

**4:DeleteQueueRear**

**5:DisplayStatus**

**6: Exit**

**Please enter your choice: 2The employee node with the ssn:1EPL is deletedDemo Double Ended Queue Operation**

**1:InsertQueueFront**

**2: DeleteQueueFront**

**3:InsertQueueRear**

**4:DeleteQueueRear**

**5:DisplayStatus**

**6: Exit**

**Please enter your choice: 4The employee node with the ssn:2EPL is deletedDemo Double Ended Queue Operation**

**1:InsertQueueFront**

**2: DeleteQueueFront**

**3:InsertQueueRear**

**4:DeleteQueueRear**

**5:DisplayStatus**

**6: Exit**

**Please enter your choice: 2Doubly Linked List is emptyDemo Double Ended Queue Operation**

**1:InsertQueueFront**

**2: DeleteQueueFront**

**3:InsertQueueRear**

**4:DeleteQueueRear**

**5:DisplayStatus**

**6: Exit**

**Please enter your choice: 6~~~Menu~~~**

**1:Create DLL of Employee Nodes**

**2:DisplayStatus**

**3:InsertAtEnd**

**4:DeleteAtEnd**

**5:InsertAtFront**

**6:DeleteAtFront**

**7:Double Ended Queue Demo using DLL**

**8:ExitPlease enter your choice: 8**

**PROGRAM-9**

**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**

**Algorithm:**

1. **Start**
2. Create header nodes head, head1, head2, head3 and link each to itself
3. Display main menu with options: 1) Evaluate polynomial, 2) Sum two polynomials, 3) Exit
4. Read user choice
5. If choice is 1:
6. Call read\_poly(head) to input polynomial P(x,y,z)
7. Call display(head) to show the polynomial
8. Call poly\_evaluate(head) to compute result by substituting x, y, z
9. Print result of evaluation
10. If choice is 2:
11. Call read\_poly(head1) to input POLY1(x,y,z)
12. Call display(head1) to display POLY1
13. Call read\_poly(head2) to input POLY2(x,y,z)
14. Call display(head2) to display POLY2
15. Call poly\_sum(head1, head2, head3) to compute POLYSUM
16. Call display(head3) to display the sum
17. If choice is 3: exit program
18. If invalid choice: show error and repeat menu
19. **Stop**

**Program:**

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#define COMPARE(x, y)((x == y) ? 0 : (x > y) ? 1 : -1)
struct node {
int coef;
int xexp, yexp, zexp;
struct node * link;
};
typedef struct node * NODE;
NODE getnode() {
NODE x;
x = (NODE) malloc(sizeof(struct node));
if (x == NULL) {
printf("Running out of memory \n");
```

```
return NULL;
}
return x;
}NODE attach(int coef, int xexp, int yexp, int zexp, NODE head) {
NODE temp, cur;
temp = getnode();
temp -> coef = coef;
temp -> xexp = xexp;
temp -> yexp = yexp;
temp -> zexp = zexp;
cur = head -> link;
while (cur -> link != head) {
cur = cur -> link;
}
cur -> link = temp; temp -> link = head; return head;
}NODE read_poly(NODE head) {
int i, j, coef, xexp, yexp, zexp, n;
printf("\nEnter the no of terms in the polynomial: ");
scanf("%d", & n);
for (i = 1; i <= n; i++) {
printf("\n\tEnter the %d term: ", i);
printf("\n\t\tCoef = ");
scanf("%d", & coef);
printf("\n\t\tEnter Pow(x) Pow(y) and Pow(z): ");
scanf("%d", & xexp);
scanf("%d", & yexp);
scanf("%d", & zexp);
head = attach(coef, xexp, yexp, zexp, head);
}
return head;
}void display(NODE head) {
NODE temp;
if (head -> link == head) {
printf("\nPolynomial does not exist.");
return;
}
temp = head -> link;while (temp != head) {
printf("%dx^%dy^%dz^%d", temp -> coef, temp -> xexp, temp -> yexp, temp -
> zexp);
temp = temp -> link;
```

```
if (temp != head)
printf(" + ");
}
}int poly_evaluate(NODE head) {
int x, y, z, sum = 0;
NODE poly;printf("\nEnter the value of x,y and z: ");
scanf("%d %d %d", & x, & y, & z);poly = head -> link;
while (poly != head) {
sum += poly -> coef * pow(x, poly -> xexp) * pow(y, poly -> yexp) * pow(z,
poly -> zexp);
poly = poly -> link;
}
return sum;
}NODE poly_sum(NODE head1, NODE head2, NODE head3) {
NODE a, b;
int coef;
a = head1 -> link;
b = head2 -> link;while (a != head1 && b != head2) {
while (1) {
if (a -> xexp == b -> xexp && a -> yexp == b -> yexp && a -> zexp == b ->
zexp) {
coef = a -> coef + b -> coef;
head3 = attach(coef, a -> xexp, a -> yexp, a -> zexp, head3);
a = a -> link;
b = b -> link;
break;
} //if ends here
if (a -> xexp != 0 || b -> xexp != 0) {
switch (COMPARE(a -> xexp, b -> xexp)) {
case -1:
head3 = attach(b -> coef, b -> xexp, b -> yexp, b -> zexp, head3);
b = b -> link;
break;case 0:
if (a -> yexp > b -> yexp) {
head3 = attach(a -> coef, a -> xexp, a -> yexp, a -> zexp, head3);
a = a -> link;
break;
} else if (a -> yexp < b -> yexp) {
head3 = attach(b -> coef, b -> xexp, b -> yexp, b -> zexp, head3);
b = b -> link;
```

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

```
if (a -> zexp != 0 || b -> zexp != 0) {
switch (COMPARE(a -> zexp, b -> zexp)) {
case -1:
head3 = attach(b -> coef, b -> xexp, b -> yexp, b -> zexp, head3);
b = b -> link;
break;
case 1:
head3 = attach(a -> coef, a -> xexp, a -> yexp, a -> zexp, head3);
a = a -> link;
break;
}
break;
}
}
}
while (a != head1) {
head3 = attach(a -> coef, a -> xexp, a -> yexp, a -> zexp, head3);
a = a -> link;
}
while (b != head2) {
head3 = attach(b -> coef, b -> xexp, b -> yexp, b -> zexp, head3);
b = b -> link;
}
return head3;
}void main() {
NODE head, head1, head2, head3;
int res, ch;
head = getnode(); /* For polynomial evaluation */
head1 = getnode(); /* To hold POLY1 */
head2 = getnode(); /* To hold POLY2 */
head3 = getnode(); /* To hold POLYSUM */head -> link = head;
head1 -> link = head1;
head2 -> link = head2;
head3 -> link = head3;while (1) {
printf("\n~~~Menu~~~");
printf("\n1.Represent and Evaluate a Polynomial P(x,y,z)");
printf("\n2.Find the sum of two polynomials POLY1(x,y,z)");
printf("\nEnter your choice:");
scanf("%d", & ch);
```

```
switch (ch) {
case 1:
printf("\n~~~~Polynomial evaluation P(x,y,z)~~~\n");
head = read_poly(head);
printf("\nRepresentation of Polynomial for evaluation: \n");
display(head);
res = poly_evaluate(head);
printf("\nResult of polynomial evaluation is : %d \n", res);
break;case 2:
printf("\nEnter the POLY1(x,y,z): \n");
head1 = read_poly(head1);
printf("\nPolynomial 1 is: \n");
display(head1);printf("\nEnter the POLY2(x,y,z): \n");
head2 = read_poly(head2);
printf("\nPolynomial 2 is: \n");
display(head2);printf("\nPolynomial addition result: \n");
head3 = poly_sum(head1, head2, head3);
display(head3);
break;
case 3:
exit(0);
}
}
}
```

### Sample Output:

~~~~Menu~~~

1.Represent and Evaluate a Polynomial P(x,y,z)

2.Find the sum of two polynomials POLY1(x,y,z) and POLY2(x,y,z)

Enter your choice: 1~~~~Polynomial evaluation P(x,y,z)~~~

Enter the no of terms in the polynomial: 5

Enter the 1 term:

Coef = 6

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

Enter the 2 term:

Coef = -4

Enter Pow(x) Pow(y) and Pow(z): 0 1 5

Enter the 3 term:

Coef = 3

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

Enter the 4 term:

Coef = 2

Enter Pow(x) Pow(y) and Pow(z): 1 5 1

Enter the 5 term:

Coef = -2

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

Representation of Polynomial for evaluation:

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

Enter the value of x,y and z: 1 1 1

Result of polynomial evaluation is : 5~~~Menu~~~

1.Represent and Evaluate a Polynomial P(x,y,z)

2.Find the sum of two polynomials POLY1(x,y,z) and POLY2(x,y,z)

Enter your choice: 2Enter the POLY1(x,y,z):

Enter the no of terms in the polynomial: 5

Enter the 1 term:

Coef = 6

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

Enter the 2 term:

Coef = 3

Enter Pow(x) Pow(y) and Pow(z): 4 3 1

Enter the 3 term:

Coef = 5

Enter Pow(x) Pow(y) and Pow(z): 0 1 1

Enter the 4 term:

Coef = 10

Enter Pow(x) Pow(y) and Pow(z): 0 1 0

Enter the 5 term:

Coef = 5

Enter Pow(x) Pow(y) and Pow(z): 0 0 0

Polynomial 1 is:

$6x^4y^4z^4 + 3x^4y^3z^1 + 5x^0y^1z^1 + 10x^0y^1z^0 +$

$5x^0y^0z^0$ **Enter the POLY2(x,y,z):**

Enter the no of terms in the polynomial: 5

Enter the 1 term:

Coef = 8

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

Enter the 2 term:

Coef = 4

Enter Pow(x) Pow(y) and Pow(z): 4 2 1

Enter the 3 term:

Coef = 30

Enter Pow(x) Pow(y) and Pow(z): 0 1 0

Enter the 4 term:

Coef = 20

Enter Pow(x) Pow(y) and Pow(z): 0 0 1

Enter the 5 term:

Coef = 3

Enter Pow(x) Pow(y) and Pow(z): 0 0 0

Polynomial 2 is:

$8x^4y^4z^4 + 4x^4y^2z^1 + 30x^0y^1z^0 + 20x^0y^0z^1 +$

$3x^0y^0z^0$ Polynomial addition result:

$14x^4y^4z^4 + 3x^4y^3z^1 + 4x^4y^2z^1 + 5x^0y^1z^1 + 40x^0y^1z^0 +$

$20x^0y^0z^1 + 8x^0y^0z^0$ ~~~Menu~~~

1. Represent and Evaluate a Polynomial P(x,y,z)

2. Find the sum of two polynomials POLY1(x,y,z) and POLY2(x,y,z)

Enter your choice:3

PROGRAM-10

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**

Algorithm:

1. **Start**
2. Display the menu with options: Create BST, Search, Traverse, Exit
3. If "Create BST" is chosen, read n values and insert each into BST (skip duplicates)
4. If "Search" is chosen, input key and search in BST using iterative traversal
5. If "BST Traversals" is chosen, display Preorder, Inorder, and Postorder traversals
6. If "Exit" is chosen, terminate the program
7. Repeat menu until "Exit" is selected
8. **Stop**

Program:

```
#include<stdio.h>
#include<stdlib.h>
struct BST {
int data;
struct BST * lchild;
struct BST * rchild;
};
typedef struct BST * NODE;NODE create() {
NODE temp;
temp = (NODE) malloc(sizeof(struct BST));
printf("\nEnter The value: ");
scanf("%d", & temp -> data);temp -> lchild = NULL;
temp -> rchild = NULL;
return temp;
}void insert(NODE root, NODE newnode);
void inorder(NODE root);
void preorder(NODE root);
void postorder(NODE root);
void search(NODE root);void insert(NODE root, NODE newnode) {
/*Note: if newnode->data == root->data it will be skipped. No duplicate nodes
are allowed */if (newnode -> data < root -> data) {
if (root -> lchild == NULL)
```

```
root -> lchild = newnode;
else
insert(root -> lchild, newnode);
}
if (newnode -> data > root -> data) {
if (root -> rchild == NULL)
root -> rchild = newnode;
else
insert(root -> rchild, newnode);
}
}void search(NODE root) {
int key;
NODE cur;
if (root == NULL) {
printf("\nBST is empty.");
return;
}
printf("\nEnter Element to be searched: ");
scanf("%d", & key);
cur = root;
while (cur != NULL) {
if (cur -> data == key) {
printf("\nKey element is present in BST");
return;
}
if (key < cur -> data)
cur = cur -> lchild;
else
cur = cur -> rchild;
}
printf("\nKey element is not found in the BST");
}void inorder(NODE root) {
if (root != NULL) {
inorder(root -> lchild);
printf("%d ", root -> data);
inorder(root -> rchild);
}
}void preorder(NODE root) {
if (root != NULL) {
printf("%d ", root -> data);
```

```
preorder(root -> lchild);
preorder(root -> rchild);
}
}void postorder(NODE root) {
if (root != NULL) {
postorder(root -> lchild);
postorder(root -> rchild);
printf("%d ", root -> data);
}
}void main() {
int ch, key, val, i, n;
NODE root = NULL, newnode;
while (1) {
printf("\n~~~~BST MENU~~~~");
printf("\n1.Create a BST");
printf("\n2.Search");
printf("\n3.BST Traversals: ");
printf("\n4.Exit");
printf("\nEnter your choice: ");
scanf("%d", & ch);
switch (ch) {
case 1:
printf("\nEnter the number of elements: ");
scanf("%d", & n);
for (i = 1; i <= n; i++) {
newnode = create();
if (root == NULL)
root = newnode;
else
insert(root, newnode);
}
break;
case 2:
if (root == NULL)
printf("\nTree Is Not Created");
else {
printf("\nThe Preorder display : ");
preorder(root);
printf("\nThe Inorder display : ");
```

```
inorder(root);
printf("\n\nThe Postorder display : ");
postorder(root);
}break;
case 3:
search(root);
break;case 4:
exit(0);
}
}
}
```

Sample output:

~~~~BST MENU~~~~

1.Create a BST

2.Search

3.BST Traversals:

4.Exit

Enter your choice: 1Enter the number of elements: 12

Enter The value: 6

Enter The value: 9

Enter The value: 5

Enter The value: 2

Enter The value: 8

Enter The value: 15

Enter The value: 24

Enter The value: 14

Enter The value: 7

Enter The value: 8

Enter The value: 5

Enter The value: 2~~~~BST MENU~~~~

1.Create a BST

2.Search

3.BST Traversals:

4.Exit



**PROGRAM-11**

**11. Develop a Program in C for the following operations on Graph(G) of Cities**

**a. Create a Graph of N cities using Adjacency Matrix.**

**b. Print all the nodes reachable from a given starting node in a digraph using DFS/BFS method**

**Algorithm:**

1. **Start**
2. Read the number of vertices  $n$
3. Read the  $n \times n$  adjacency matrix for the graph
4. Initialize `visited[]` array to 0 for all vertices
5. Read the starting vertex `start`
6. Display menu: 1. BFS 2. DFS 3. Exit
7. If choice is 1 (BFS), call `bfs(start)` to print reachable nodes
8. If choice is 2 (DFS), call `dfs(start)` to print reachable nodes
9. If choice is 3, exit the program
10. If choice is invalid, prompt for a valid choice
11. **Stop**

**Program:**

```
#include<stdio.h>#include<stdlib.h>int a[50][50], n, visited[50];
int q[20], front = -1, rear = -1;
int s[20], top = -1, count = 0;void bfs(int v) {
int i, cur;
visited[v] = 1;
q[++rear] = v;
while (front != rear) {
cur = q[++front];
for (i = 1; i <= n; i++) {
if ((a[cur][i] == 1) && (visited[i] == 0)) {
q[++rear] = i;
visited[i] = 1;
printf("%d ", i);
}
}
}
}void dfs(int v) {
int i;
```

```
visited[v] = 1;
s[++top] = v;
for (i = 1; i <= n; i++) {
if (a[v][i] == 1 && visited[i] == 0) {
printf("%d ", i);
dfs(i);
}
}
}int main() {int ch, start, i, j;
printf("\nEnter the number of vertices in graph:");
scanf("%d", & n);
printf("\nEnter the adjacency matrix:\n");
for (i = 1; i <= n; i++) {
for (j = 1; j <= n; j++)
scanf("%d", & a[i][j]);
}for (i = 1; i <= n; i++)
visited[i] = 0;
printf("\nEnter the starting vertex: ");
scanf("%d", & start);printf("\n==>1. BFS: Print all nodes reachable from a
given starting node");
printf("\n==>2. DFS: Print all nodes reachable from a given starting node");
printf("\n==>3:Exit");
printf("\nEnter your choice: ");
scanf("%d", & ch);
switch (ch) {
case 1:
printf("\nNodes reachable from starting vertex %d are: ", start);
bfs(start);
for (i = 1; i <= n; i++) {
if (visited[i] == 0)
printf("\nThe vertex that is not reachable is %d", i);
}
break;case 2:
printf("\nNodes reachable from starting vertex %d are:\n", start);
dfs(start);
break;
case 3:
exit(0);
default:
```

```
printf("\nPlease enter valid choice:");  
}  
}
```

### Sample Output:

\*\*\*\*\*case-1\*\*\*\*\*

Enter the number of vertices in graph:4

Enter the adjacency matrix:

0 1 0 1

0 0 1 0

0 0 0 1

0 0 0 0

Enter the starting vertex: 1

==>1. BFS: Print all nodes reachable from a given starting node

==>2. DFS: Print all nodes reachable from a given starting node

==>3:Exit

Enter your choice: 1

Nodes reachable from starting vertex 1 are: 2 4

3\*\*\*\*\*case-2\*\*\*\*\*Enter the

number of vertices in graph:4

Enter the adjacency matrix:

0 1 0 1

0 0 1 0

0 0 0 1

0 0 0 0

Enter the starting vertex: 2

==>1. BFS: Print all nodes reachable from a given starting node

==>2. DFS: Print all nodes reachable from a given starting node

==>3:Exit

Enter your choice: 1

Nodes reachable from starting vertex 2 are: 3 4

The vertex that is not reachable is 1\*\*\*\*\*case-

3\*\*\*\*\*Enter the number of vertices in graph:4

Enter the adjacency matrix:

0 1 0 1

0 0 1 0

0 0 0 1

0 0 0 0

Enter the starting vertex: 1

==>1. BFS: Print all nodes reachable from a given starting node

==>2. DFS: Print all nodes reachable from a given starting node

==>3:Exit

Enter your choice: 2

Nodes reachable from starting vertex 1 are: 2 3

4\*\*\*\*\*case-4\*\*\*\*\*Enter the  
number of vertices in graph:4

Enter the adjacency matrix:

0 1 0 1

0 0 1 0

0 0 0 1

0 0 0 0

Enter the starting vertex: 2

==>1. BFS: Print all nodes reachable from a given starting node

==>2. DFS: Print all nodes reachable from a given starting node

==>3:Exit

Enter your choice: 2

Nodes reachable from starting vertex 2 are: 3 4

**PROGRAM-12**

**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 Hashfunction  $H: K \rightarrow L$  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.**

**Algorithm:**

1. **Start**
2. Read the number of employee records n
3. Read the hash table size m (2-digit memory locations)
4. Allocate a hash table of size m and initialize all entries to -1
5. Read n four-digit keys
6. For each key, compute index using  $index = key \% m$
7. If collision occurs, resolve it using linear probing:  $index = (index + 1) \% m$  until an empty slot is found
8. Insert the key at the calculated index and increment count
9. If count reaches m, stop inserting and show overflow message
10. Display the contents of the hash table
11. **Stop**

**Program**

```
#include<stdio.h>#include<stdlib.h>int key[20], n, m;
int * ht, index;
int count = 0;void insert(int key) {
index = key % m;
while (ht[index] != -1) {
index = (index + 1) % m;
}
ht[index] = key;
count++;
}void display() {
int i;
if (count == 0) {
printf("\nHash Table is empty");
return;
}printf("\nHash Table contents are:\n ");
```

```
for (i = 0; i < m; i++)
printf("\n T[%d] --> %d ", i, ht[i]);
}void main() {
int i;
printf("\nEnter the number of employee records (N) :");
scanf("%d", & n);printf("\nEnter the two digit memory locations (m) for hash
table:");
scanf("%d", & m);ht = (int * ) malloc(m * sizeof(int));
for (i = 0; i < m; i++)
ht[i] = -1;printf("\nEnter the four digit key values (K) for N Employee
Records:\n ");
for (i = 0; i < n; i++)
scanf("%d", & key[i]);for (i = 0; i < n; i++) {
if (count == m) {
printf("\n~~~Hash table is full. Cannot insert the record %d key~~~", i + 1);
break;
}
insert(key[i]);
} //Displaying Keys inserted into hash table
display();
}
```

**Sample Output :**

```
Enter the number of employee records (N) :10
Enter the two digit memory locations (m) for hash table:15
Enter the four digit key values (K) for N Employee Records:
4020
4560
9908
6785
0423
7890
6547
3342
9043
6754
```

Hash Table contents are:

T[0] --> 4020

T[1] --> 4560

T[2] --> 7890

T[3] --> 423

T[4] --> 6754

T[5] --> 6785

T[6] --> -1

T[7] --> 6547

T[8] --> 9908

T[9] --> -1

T[10] --> -1

T[11] --> -1

T[12] --> 3342

T[13] --> 9043

T[14] --> -1