Sri SAIRAM COLLEGE OF ENGINEERING
Anekal, Bengaluru

Accredited by NAAC
ISO 9001:2015 Certified Institution
Approved by AICTE, New Delhi
Affiliated to Visvesvaraya Technological University
www.sairamce.edu.in

DEPARTMENT OF INFORMATION SCIENCE AND ENGINEERING

SEMESTER-III

DATA STRUCTURES & APPLICATIONS (BCSL305)

LAB MANUAL

Scheme 2022

Academic Year (2023-2024)

Prepared by,

Prof. Kusuma J

Assistant professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Sri Sairam College of Engineering

Program 1:

Develop a Program in C for the following:

a) Declare a calendar as an array of 7 elements (A dynamically Created array) to represent 7 days of a week. Each Element of the array is a structure having three fields. The first field is the name of the Day (A dynamically allocated String), The second field is the date of the Day (A integer), the third field is the description of the activity for a particular day (A dynamically allocated String).

b)Write functions create(), read() and display(); to create the calendar, to read the data from the keyboard and to print weeks activity details report on screen.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
// Define the structure for a day
struct Day {
   char* name;        // Dynamically allocated string for the day's name
   int date;
   char* activity;    // Dynamically allocated string for the day's activity
};

// Function to create the calendar
void create(struct Day calendar[7])
{
   for (int i = 0; i < 7; i++) {
      calendar[i].name = NULL;
      calendar[i].activity = NULL;
   }
}
// Function to read data from the keyboard
void read(struct Day calendar[7]) {
   for (int i = 0; i < 7; i++) {
      calendar[i].name = (char*)malloc(20 * sizeof(char));  // Allocate memory for the day's name
      calendar[i].activity = (char*)malloc(100 * sizeof(char)); // Allocate memory for the activity
                                                                description
      printf("Enter the name of day %d: ", i + 1);
      scanf("%s", calendar[i].name);
      printf("Enter the date for %s: ", calendar[i].name);
      scanf("%d", &calendar[i].date);
      printf("Enter the activity for %s: ", calendar[i].name);
      scanf("%s", calendar[i].activity);
      printf("\n");
```

```
    }
}

// Function to display the calendar
void display(struct Day calendar[7]) {
    printf("\nCalendar for the week:\n\n");
    for (int i = 0; i < 7; i++)
{
        printf("Day %d: %s (Date: %d) - Activity: %s\n\n", i + 1, calendar[i].name, calendar[i].date,
calendar[i].activity);
    }
}

// Function to free allocated memory
void freeMemory(struct Day calendar[7]) {
    for (int i = 0; i < 7; i++) {
        free(calendar[i].name);
        free(calendar[i].activity);
    }
}
int main() {
    struct Day calendar[7];
    // Create the calendar
    create(calendar);
    // Read data from the keyboard
    read(calendar);
    // Display the calendar
    display(calendar);
    // Free allocated memory
    freeMemory(calendar);
    return 0;
}
```

Output:

```
Enter the name of day 1: Monday
Enter the date for Monday: 1
Enter the activity for Monday: Meeting

Enter the name of day 2: Tuesday
Enter the date for Tuesday: 2
Enter the activity for Tuesday: Gym
```

Enter the name of day 3: Wednesday
Enter the date for Wednesday: 3
Enter the activity for Wednesday: Coding

Enter the name of day 4: Thursday
Enter the date for Thursday: 4
Enter the activity for Thursday: Shopping

Enter the name of day 5: Friday
Enter the date for Friday: 5
Enter the activity for Friday: Movie

Enter the name of day 6: Saturday
Enter the date for Saturday: 6
Enter the activity for Saturday: Picnic

Enter the name of day 7: Sunday
Enter the date for Sunday: 7
Enter the activity for Sunday: Reading

Calendar for the week:

Day 1: Monday (Date: 1) - Activity: Meeting

Day 2: Tuesday (Date: 2) - Activity: Gym

Day 3: Wednesday (Date: 3) - Activity: Coding

Day 4: Thursday (Date: 4) - Activity: Shopping

Day 5: Friday (Date: 5) - Activity: Movie

Day 6: Saturday (Date: 6) - Activity: Picnic

Day 7: Sunday (Date: 7) - Activity: Reading

Program 2:

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

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

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

Output:

Output test case 1

```
Enter the main string:this is me
Enter the pattern string:me
Enter the replace string:you
Resultant string is this is you
```

Output test case 2

```
Enter the main string:this is me
Enter the pattern string:are
Enter the replace string:not
Pattern string is not found
```

Program 3:

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

a. Push an Element on to Stack
b. Pop an Element from Stack
c. Demonstrate how Stack can be used to check Palindrome
d. Demonstrate Overflow and Underflow situations on Stack
e. Display the status of Stack
f. Exit
Support the program with appropriate functions for each of the above operations

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h> // Include string.h for string operations
#define MAX 10
int stack_arr[MAX];
int top = -1;

void push(int item);
int pop();
int peek();
int isEmpty();
int isFull();
void display();
int isPalindrome(char str[]); // Function to check if a string is a palindrome

int main() {
   int choice, item;
   char str[MAX];
   while (1) {
      printf("\n1.Push\n");
      printf("2.Pop\n");
      printf("3.Display the top element\n");
      printf("4.Display all stack elements\n");
      printf("5.Check Palindrome\n"); // Option to check a string for palindrome
      printf("6.Quit\n");
      printf("\nEnter your choice: ");
      scanf("%d", &choice);
      switch (choice) {
      case 1:
         printf("\nEnter the item to be pushed: ");
```

```
            scanf("%d", &item);
            push(item);
            break;
        case 2:
            item = pop();
            printf("\nPopped item is: %d\n", item);
            break;
        case 3:
            printf("\nItem at the top is: %d\n", peek());
            break;
        case 4:
            display();
            break;
        case 5:
            printf("\nEnter a string to check if it's a palindrome: ");
            scanf("%s", str);
            if (isPalindrome(str)) {
                printf("%s is a palindrome.\n", str);
            } else {
                printf("%s is not a palindrome.\n", str);
            }break;
        case 6:
            exit(0);
        default:
            printf("\nWrong choice\n");
        } /*End of switch*/
    }     /*End of while*/
  return 0;
} /*End of main()*/

void push(int item) {
    if (isFull()) {
        printf("\nStack Overflow\n");
        return;
    }
    top = top + 1;
    stack_arr[top] = item;
} /*End of push()*/
int pop() {
    int item;
    if (isEmpty()) {
```

```
        printf("\nStack Underflow\n");
        exit(1);
    }
    item = stack_arr[top];
    top = top - 1;
    return item;
} /*End of pop()*/

int peek() {
    if (isEmpty()) {
        printf("\nStack Underflow\n");
        exit(1);
    }
    return stack_arr[top];
} /*End of peek()*/

int isEmpty() {
    if (top == -1)
        return 1;
    else
        return 0;
} /*End of isEmpty*/

int isFull() {
    if (top == MAX - 1)
        return 1;
    else
        return 0;
} /*End of isFull()*/
void display() {
    int i;
    if (isEmpty()) {
        printf("\nStack is empty\n");
        return;
    }
    printf("\nStack elements:\n\n");
    for (i = top; i >= 0; i--)
        printf(" %d\n", stack_arr[i]);
    printf("\n");
} /*End of display()*/
```

```
int isPalindrome(char str[]) {
    int i, len;
    len = strlen(str);

    for (i = 0; i < len; i++) {
        push(str[i]); // Push characters of the string onto the stack
    }

    for (i = 0; i < len; i++) {
        char ch = pop();
        if (ch != str[i]) {
            return 0; // Not a palindrome
        }
    }
    return 1; // Palindrome
}
```

Output:

```
1.Push
2.Pop
3.Display the top element
4.Display all stack elements
5.Check Palindrome
6.Quit

Enter your choice: 1
Enter the item to be pushed: 20
1.Push
2.Pop
3.Display the top element
4.Display all stack elements
5.Check Palindrome
6.Quit
Enter your choice: 1
Enter the item to be pushed: 10
1.Push
2.Pop
3.Display the top element
4.Display all stack elements
5.Check Palindrome
6.Quit
```

Enter your choice: 1
Enter the item to be pushed: 30
1.Push
2.Pop
3.Display the top element
4.Display all stack elements
5.Check Palindrome
6.Quit

Enter your choice: 5
Enter a string to check if it's a palindrome: madam
madam is a palindrome.

1.Push
2.Pop
3.Display the top element
4.Display all stack elements
5.Check Palindrome
6.Quit

Enter your choice: 3
Item at the top is: 30

1.Push
2.Pop
3.Display the top element
4.Display all stack elements
5.Check Palindrome
6.Quit

Enter your choice: 4
Stack elements:

30
10
20

1.Push
2.Pop

3.Display the top element
4.Display all stack elements
5.Check Palindrome
6.Quit

Enter your choice: 2
Popped item is: 30

1.Push
2.Pop
3.Display the top element
4.Display all stack elements
5.Check Palindrome
6.Quit

Enter your choice: 6

Program 4:

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

```
#include<stdio.h>
#include<ctype.h>
char stack[100];
int top = -1;
void push(char x)
{
   stack[++top] = x;
}

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

int priority(char x)
{
   if(x == '(')
      return 0;
   if(x == '+' || x == '-')
      return 1;
   if(x == '*' || x == '/')
      return 2;
   return 0;
}

int main()
{
   char exp[100];
   char *e, x;
   printf("Enter the expression : ");
   scanf("%s",exp);
   printf("\n");
   e = exp;
```

```
    while(*e != '\0')
   {
      if(isalnum(*e))
         printf("%c ",*e);
      else if(*e == '(')
         push(*e);
      else if(*e == ')')
      {
         while((x = pop()) != '(')
            printf("%c ", x);
      }
      else
      {
         while(priority(stack[top]) >= priority(*e))
            printf("%c ",pop());
         push(*e);
      }
      e++;
   }

   while(top != -1)
   {
      printf("%c ",pop());
   }return 0;
}
```

Output:

Output Test Case 1:
Enter the expression : a+b*c

a b c * +

Output Test Case 2:
Enter the expression : (a+b)*c+(d-a)

a b + c * d a - +

Output Test Case 3:
Enter the expression : ((4+8)(6-5))/((3-2)(2+2))
4 8 + 6 5 - 3 2 - 2 2 + /

Program 5 :

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

```
 #include<stdio.h>
#include<stdlib.h>
#include<math.h>
int i, top = -1;
int op1, op2, res, s[20];
char postfix[90], symb;

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

int pop()
{
        int item;
        item  =  s[top];
        top = top-1;
        return item;
}

void main()
{
        printf("\nEnter a valid postfix expression:\n");
        scanf("%s", postfix);
        for(i=0; postfix[i]!='\0'; i++)
        {
                symb = postfix[i];
                if(isdigit(symb))
                {
                        push(symb - '0');
                }
                else
                {
                        op2 = pop();
                        op1 = pop();
                        switch(symb)
                        {
```

```
                        case '+':       push(op1+op2);
                                        break;
                        case '-':       push(op1-op2);
                                        break;
                        case '*':       push(op1*op2);
                                        break;
                        case '/':       push(op1/op2);
                                        break;
                        case '%':       push(op1%op2);
                                        break;
                        case '$':
                        case '^':       push(pow(op1, op2));
                                        break;
                        default :   push(0);
                    }
                }
        }
        res = pop();
        printf("\n Result = %d", res);
}
```

Output:

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

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

Program 5 :

(b) Solving Tower of Hanoi problem with n disks

```
#include <stdio.h>
void towerOfHanoi(int n, char source, char auxiliary, char destination)
{
   if (n == 1)
{
      printf("Move disk 1 from %c to %c\n", source, destination);
      return;
   }

   // Move n-1 disks from source to auxiliary peg using destination as temporary peg
   towerOfHanoi(n - 1, source, destination, auxiliary);

   // Move the nth disk from source to destination peg
   printf("Move disk %d from %c to %c\n", n, source, destination);

   // Move the n-1 disks from auxiliary peg to destination peg using source as temporary peg
   towerOfHanoi(n - 1, auxiliary, source, destination);
}

int main() {
   int n;
   printf("Enter the number of disks: ");
   scanf("%d", &n);
    printf("Tower of Hanoi solution for %d disks:\n", n);
   towerOfHanoi(n, 'A', 'B', 'C'); // A, B, and C represent the three pegs
    return 0;
}
```

Output:

```
Enter the number of disks: 3
Tower of Hanoi solution for 3 disks:
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:

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

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 10
int queue_arr[MAX];
int rear=-1;
int front=-1;
void insert(int item);
int del();
int peek();
void display();
int isFull();
int isEmpty();
int main()
{
    int choice,item;
    while(1)
    {
        printf("\n1.Insert\n");
        printf("2.Delete\n");
        printf("3.Display element at the front\n");
        printf("4.Display all elements of the queue\n");
        printf("5.Quit\n");
        printf("\nEnter your choice : ");
        scanf("%d",&choice);

        switch(choice)
        {
        case 1:
            printf("\nInput the element for adding in queue : ");
            scanf("%d",&item);
```

```
                insert(item);
                break;
        case 2:
                item=del();
                printf("\nDeleted element is  %d\n",item);
                break;
        case 3:
                printf("\nElement at the front is %d\n",peek());
                break;
        case 4:
                display();
                break;
        case 5:
                exit(1);
        default:
                printf("\nWrong choice\n");
        }/*End of switch*/
    }/*End of while*/
    return 0;
}/*End of main()*/

void insert(int item)
{
    if( isFull() )
    {
        printf("\nQueue Overflow\n");
        return;
    }
    if( front == -1 )
        front=0;
    rear=rear+1;
    queue_arr[rear]=item ;
}/*End of insert()*/

int del()
{
    int item;
    if( isEmpty() )
    {
        printf("\nQueue Underflow\n");
        exit(1);
```

```
        }
        item=queue_arr[front];
        front=front+1;
        return item;
}/*End of del()*/

int peek()
{
        if( isEmpty() )
        {
                printf("\nQueue Underflow\n");
                exit(1);
        }
        return queue_arr[front];
}/*End of peek()*/

int isEmpty()
{
        if( front==-1 || front==rear+1 )
                return 1;
        else
                return 0;
}/*End of isEmpty()*/

int isFull()
{
        if( rear==MAX-1 )
                return 1;
        else
                return 0;
}/*End of isFull()*/

void display()
{
        int i;
        if ( isEmpty() )
        {
                printf("\nQueue is empty\n");
                return;
        }
        printf("\nQueue is :\n\n");
```

```
    for(i=front;i<=rear;i++)
        printf("%d  ",queue_arr[i]);
    printf("\n\n");
}
```

Output:

1.Insert

2.Delete

3.Display element at the front

4.Display all elements of the queue

5.Quit

Enter your choice : 1

Input the element for adding in queue : 1

1.Insert

2.Delete

3.Display element at the front

4.Display all elements of the queue

5.Quit

Enter your choice : 1

Input the element for adding in queue : 2

1.Insert

2.Delete

3.Display element at the front

4.Display all elements of the queue

5.Quit

Enter your choice : 1

Input the element for adding in queue : 3

1.Insert

2.Delete

3.Display element at the front

4.Display all elements of the queue

5.Quit

Enter your choice : 1

Input the element for adding in queue : 4

1.Insert

2.Delete

3.Display element at the front

4.Display all elements of the queue

5.Quit

Enter your choice : 4

Queue is :

1 2 3 4

1.Insert

2.Delete

3.Display element at the front

4.Display all elements of the queue

5.Quit

Enter your choice : 3

Element at the front is 1

1.Insert

2.Delete

3.Display element at the front

4.Display all elements of the queue

5.Quit

Enter your choice : 2

Deleted element is 1

1.Insert

2.Delete

3.Display element at the front

4.Display all elements of the queue

5.Quit

Enter your choice : 2

Deleted element is 2

1.Insert

2.Delete

3.Display element at the front

4.Display all elements of the queue

5.Quit

Enter your choice : 2

Deleted element is 3

1.Insert

2.Delete

3.Display element at the front

4.Display all elements of the queue

5.Quit

Enter your choice : 2

Deleted element is 4

1.Insert

2.Delete

3.Display element at the front

4.Display all elements of the queue

5.Quit

Enter your choice : 2

Queue Underflow

Program 7:

Develop a menu driven Program in C for the following operations on Singly Linked List (SLL) of Student Data with the fields: USN, Name, Programme, Sem, PhNo

a. Create a SLL of N Students Data by using front insertion.
b. Display the status of SLL and count the number of nodes in it
c. Perform Insertion / Deletion at End of SLL
d. Perform Insertion / Deletion at Front of SLL(Demonstration of stack)
e. Exit

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
// Structure to represent student data
struct Student {
    char usn[20];
    char name[50];
    char program[50];
    int sem;
    long long int phNo;
    struct Student *next;
};
typedef struct Student Student;
Student *head = NULL;

// Function to create a new student node
Student *createStudent() {
    Student *newStudent = (Student *)malloc(sizeof(Student));
    if (newStudent == NULL) {
        printf("Memory allocation failed.\n");
        exit(1);
    }
    printf("Enter USN: ");
    scanf("%s", newStudent->usn);
    printf("Enter Name: ");
    scanf("%s", newStudent->name);
    printf("Enter Program: ");
    scanf("%s", newStudent->program);
    printf("Enter Semester: ");
    scanf("%d", &newStudent->sem);
    printf("Enter Phone Number: ");
```

```
    scanf("%lld", &newStudent->phNo);
    newStudent->next = NULL;
    return newStudent;
}

// Function to insert a student at the front of the list
void insertFront() {
    Student *newStudent = createStudent();
    newStudent->next = head;
    head = newStudent;
    printf("Student added at the front.\n");
}

// Function to insert a student at the end of the list
void insertEnd() {
    Student *newStudent = createStudent();
    if (head == NULL) {
        head = newStudent;
    } else {
        Student *current = head;
        while (current->next != NULL) {
            current = current->next;
        }
        current->next = newStudent;
    }
    printf("Student added at the end.\n");
}

// Function to delete a student from the front of the list (stack-like operation)
void deleteFront() {
    if (head == NULL) {
        printf("List is empty, cannot delete.\n");
    } else {
        Student *temp = head;
        head = head->next;
        free(temp);
        printf("Student deleted from the front.\n");
    }
}

// Function to display the status of the linked list and count the number of nodes
```

```
void displayAndCount() {
    if (head == NULL) {
        printf("List is empty.\n");
    } else {
        Student *current = head;
        int count = 0;
        printf("Student Data:\n");
        while (current != NULL) {
            printf("USN: %s, Name: %s, Program: %s, Semester: %d, Phone: %lld\n",
                   current->usn, current->name, current->program, current->sem, current->phNo);
            current = current->next;
            count++;
        }
        printf("Total Students: %d\n", count);
    }
}

int main() {
    int choice;
    while (1) {
        printf("\nMenu:\n");
        printf("1. Insert at Front\n");
        printf("2. Insert at End\n");
        printf("3. Delete from Front\n");
        printf("4. Display and Count\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                insertFront();
                break;
            case 2:
                insertEnd();
                break;
            case 3:
                deleteFront();
                break;
            case 4:
                displayAndCount();
```

```
            break;
        case 5:
            exit(0);
        default:
            printf("Invalid choice. Please try again.\n");
      }
   }

   return 0;
}
```

Output:

Menu:

1. Insert at Front

2. Insert at End

3. Delete from Front

4. Display and Count

5. Exit

Enter your choice: 1

Enter USN: 1234

Enter Name: John

Enter Program: Engineering

Enter Semester: 2

Enter Phone Number: 1234567890

Student added at the front.

Menu:

1. Insert at Front

2. Insert at End

3. Delete from Front

4. Display and Count

5. Exit

Enter your choice: 2

Enter USN: 5678

Enter Name: Alice

Enter Program: Computer Science

Enter Semester: 3

Enter Phone Number: 9876543210

Student added at the end.

Menu:

1. Insert at Front

2. Insert at End

3. Delete from Front

4. Display and Count

5. Exit

Enter your choice: 4

Student Data:

USN: 1234, Name: John, Program: Engineering, Semester: 2, Phone: 1234567890

USN: 5678, Name: Alice, Program: Computer Science, Semester: 3, Phone: 9876543210

Total Students: 2

Menu:

1. Insert at Front

2. Insert at End

3. Delete from Front

4. Display and Count

5. Exit

Enter your choice: 3

Student deleted from the front.

Menu:

1. Insert at Front

2. Insert at End

3. Delete from Front

4. Display and Count

5. Exit

Enter your choice: 4

Student Data:

USN: 5678, Name: Alice, Program: Computer Science, Semester: 3, Phone: 9876543210

Total Students: 1

Menu:

1. Insert at Front

2. Insert at End

3. Delete from Front

4. Display and Count

5. Exit

Enter your choice: 5

Program 8:

Develop a menu driven Program in C for the following operations on Doubly Linked List (DLL) of Employee Data with the fields: SSN, Name, Dept, Designation, Sal, PhNo

a. Create a DLL of N Employees Data by using end insertion.
b. Display the status of DLL and count the number of nodes in it
c. Perform Insertion and Deletion at End of DLL
d. Perform Insertion and Deletion at Front of DLL
e. Demonstrate how this DLL can be used as Double Ended Queue.
f. Exit

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
// Structure to represent employee data
struct Employee {
    char ssn[20];
    char name[50];
    char dept[50];
    char designation[50];
    float sal;
    long long int phNo;
    struct Employee *prev;
    struct Employee *next;
};

typedef struct Employee Employee;
Employee *head = NULL;
Employee *tail = NULL;
// Function to create a new employee node
Employee *createEmployee() {
    Employee *newEmployee = (Employee *)malloc(sizeof(Employee));
    if (newEmployee == NULL) {
        printf("Memory allocation failed.\n");
        exit(1);
    }

    printf("Enter SSN: ");
    scanf("%s", newEmployee->ssn);
    printf("Enter Name: ");
```

```
    scanf("%s", newEmployee->name);
    printf("Enter Department: ");
    scanf("%s", newEmployee->dept);
    printf("Enter Designation: ");
    scanf("%s", newEmployee->designation);
    printf("Enter Salary: ");
    scanf("%f", &newEmployee->sal);
    printf("Enter Phone Number: ");
    scanf("%lld", &newEmployee->phNo);
    newEmployee->prev = NULL;
    newEmployee->next = NULL;
    return newEmployee;
}

// Function to insert an employee at the end of the list
void insertEnd() {
    Employee *newEmployee = createEmployee();
    if (head == NULL) {
        head = newEmployee;
        tail = newEmployee;
    } else {
        newEmployee->prev = tail;
        tail->next = newEmployee;
        tail = newEmployee;
    }
    printf("Employee added at the end.\n");
}

// Function to delete an employee from the end of the list
void deleteEnd() {
    if (head == NULL) {
        printf("List is empty, cannot delete.\n");
    } else if (head == tail) {
        free(head);
        head = NULL;
        tail = NULL;
        printf("Employee deleted from the end.\n");
    } else {
        Employee *temp = tail;
        tail = tail->prev;
        tail->next = NULL;
```

```
        free(temp);
        printf("Employee deleted from the end.\n");
    }
}

// Function to insert an employee at the front of the list
void insertFront() {
    Employee *newEmployee = createEmployee();
    if (head == NULL) {
        head = newEmployee;
        tail = newEmployee;
    } else {
        newEmployee->next = head;
        head->prev = newEmployee;
        head = newEmployee;
    }
    printf("Employee added at the front.\n");
}

// Function to delete an employee from the front of the list
void deleteFront() {
    if (head == NULL) {
        printf("List is empty, cannot delete.\n");
    } else if (head == tail) {
        free(head);
        head = NULL;
        tail = NULL;
        printf("Employee deleted from the front.\n");
    } else {
        Employee *temp = head;
        head = head->next;
        head->prev = NULL;
        free(temp);
        printf("Employee deleted from the front.\n");
    }
}

// Function to display the status of the linked list and count the number of nodes
void displayAndCount() {
    if (head == NULL) {
        printf("List is empty.\n");
```

```
    } else {
        Employee *current = head;
        int count = 0;
        printf("Employee Data:\n");
        while (current != NULL) {
            printf("SSN: %s, Name: %s, Department: %s, Designation: %s, Salary: %.2f, Phone: %lld\n",
                   current->ssn, current->name, current->dept, current->designation, current->sal, current->phNo);
            current = current->next;
            count++;
        }
        printf("Total Employees: %d\n", count);
    }
}

int main() {
    int choice;
    while (1) {
        printf("\nMenu:\n");
        printf("1. Insert at End\n");
        printf("2. Delete from End\n");
        printf("3. Insert at Front\n");
        printf("4. Delete from Front\n");
        printf("5. Display and Count\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                insertEnd();
                break;
            case 2:
                deleteEnd();
                break;
            case 3:
                insertFront();
                break;
            case 4:
                deleteFront();
```

```
            break;
        case 5:
            displayAndCount();
            break;
        case 6:
            exit(0);
        default:
            printf("Invalid choice. Please try again.\n");
      }
   }

   return 0;
}
```

Output:

```
Menu:
1. Insert at End
2. Delete from End
3. Insert at Front
4. Delete from Front
5. Display and Count
6. Exit
Enter your choice: 1

Enter SSN: 12345
Enter Name: John
Enter Department: HR
Enter Designation: Manager
Enter Salary: 60000
Enter Phone Number: 1234567890
Employee added at the end.

Menu:
1. Insert at End
2. Delete from End
3. Insert at Front
4. Delete from Front
5. Display and Count
6. Exit
Enter your choice: 1
```

Enter SSN: 67890
Enter Name: Alice
Enter Department: IT
Enter Designation: Engineer
Enter Salary: 55000
Enter Phone Number: 9876543210
Employee added at the end.

Menu:
1. Insert at End
2. Delete from End
3. Insert at Front
4. Delete from Front
5. Display and Count
6. Exit
Enter your choice: 3

Enter SSN: 11111
Enter Name: Bob
Enter Department: Sales
Enter Designation: Executive
Enter Salary: 75000
Enter Phone Number: 5555555555
Employee added at the front.

Menu:
1. Insert at End
2. Delete from End
3. Insert at Front
4. Delete from Front
5. Display and Count
6. Exit
Enter your choice: 5

Employee Data:
SSN: 11111, Name: Bob, Department: Sales, Designation: Executive, Salary: 75000.00, Phone: 5555555555
SSN: 12345, Name: John, Department: HR, Designation: Manager, Salary: 60000.00, Phone: 1234567890
SSN: 67890, Name: Alice, Department: IT, Designation: Engineer, Salary: 55000.00, Phone: 9876543210

Total Employees: 3

Menu:
1. Insert at End
2. Delete from End
3. Insert at Front
4. Delete from Front
5. Display and Count
6. Exit
Enter your choice: 4

Employee deleted from the front.

Menu:
1. Insert at End
2. Delete from End
3. Insert at Front
4. Delete from Front
5. Display and Count
6. Exit
Enter your choice: 2

Employee deleted from the end.

Menu:
1. Insert at End
2. Delete from End
3. Insert at Front
4. Delete from Front
5. Display and Count
6. Exit
Enter your choice: 5

Employee Data:
SSN: 12345, Name: John, Department: HR, Designation: Manager, Salary: 60000.00, Phone: 1234567890
Total Employees: 1

Menu:
1. Insert at End
2. Delete from End

3. Insert at Front
4. Delete from Front
5. Display and Count
6. Exit

Enter your choice: 6

Program 9:

Develop a Program in C for the following operations on Singly Circular Linked List (SCLL) with header nodes a. Represent and Evaluate a Polynomial P(x,y,z) = 6x 2 y 2 z-4yz 5 +3x 3 yz+2xy 5 z-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>
typedef struct polynomial
{
   float coeff;
   int x,y,z;
   struct polynomial *next;
}poly;
poly *p1,*p2,*p3;
poly* readpoly()
{
   poly *temp=(poly*)malloc(sizeof(poly));
   printf("\nEnter coeff:");
   scanf("%f",&temp->coeff);
   printf("Enter x expon:");
   scanf("%d",&temp->x);
   printf("Enter y expon:");
   scanf("%d",&temp->y);
   printf("Enter z expon:");
   scanf("%d",&temp->z);
   return temp;
}
poly* create()
{
   int n,i;
   printf("\nEnter no. of terms:");
   scanf("%d",&n);
   poly *temp=(poly*)malloc(sizeof(poly)),*t1=temp;
   for(i=0;i<n;i++,t1=t1->next)
     t1->next=readpoly();
   t1->next=temp;
   return temp;
}
```

```
void evaluate()
{
   float sum=0;
   int x,y,z;
   poly *t=p1->next;
   printf("\nEnter x,y&z:\n");
   scanf("%d",&x);
   scanf("%d",&y);
   scanf("%d",&z);
   while(t!=p1)
   {
     sum+=t->coeff*pow(x,t->x)*pow(y,t->y)*pow(z,t->z);
     t=t->next;
   }
   printf("\nSum=%f",sum);
}
void display(poly *p)
{
   poly *t=p->next;
   while(t!=p)
   {
     if(t!=p->next&&t->coeff>0)
       putchar('+');
     printf("%.1fx^%dy^%dz^%d",t->coeff,t->x,t->y,t->z);
     t=t->next;
   }
}
poly* attach(float coeff,int x,int y,int z,poly *p)
{
   poly *t=(poly*)malloc(sizeof(poly));
   t->coeff=coeff;
   t->x=x;
   t->y=y;
   t->z=z;
   p->next=t;
   return t;
}
poly* add()
{
   printf("\nPolynomial1:\n");
   p1=create();
```

```
    printf("\nPolynomial2:\n");
    p2=create();
    int flag;
    poly *t1=p1->next,*t2=p2->next,*t3;
    p3=(poly*)malloc(sizeof(poly));
    t3=p3;
    while(t1!=p1&&t2!=p2)
    {
       if(t1->x>t2->x)
         flag=1;
       else if(t1->y<t2->y)
         flag=-1;
       else if(t1->z==t2->z)
         flag=0;
       switch(flag)
       {
          case 0:t3=attach(t1->coeff+t2->coeff,t1->x,t1->y,t1->z,t3);
                 t1=t1->next;
                 t2=t2->next;
                 break;
          case 1:t3=attach(t1->coeff,t1->x,t1->y,t1->z,t3);
                 t1=t1->next;
                 break;
          case -1:t3=attach(t2->coeff,t2->x,t2->y,t2->z,t3);
                  t2=t2->next;
                  break;
       }
    }
    for(;t1!=p1;t1=t1->next)
       t3=attach(t1->coeff,t1->x,t1->y,t1->z,t3);
    for(;t2!=p2;t2=t2->next)
       t3=attach(t2->coeff,t2->x,t2->y,t2->z,t3);
    t3->next=p3;
    return p3;
}
int main()
{
   int ch;
   printf("\n1.Represent and evaluate polynomial\n2.Add 2 polynomials\n3.Exit\nEnter choice:");
   scanf("%d",&ch);
   switch(ch)
```

```
  {
     case 1:p1=create();
            display(p1);
            evaluate();
            break;
     case 2:p3=add();
            printf("\nPolynomial1:\n");
            display(p1);
            printf("\nPolynomial2:\n");
            display(p2);
            printf("\nP1+P2:\n");
            display(p3);
            break;
     case 3:exit(0);
     default:printf("\nInvalid choice...!");
  }
   return 0;

}
```

Output:

```
1. Represent and evaluate polynomial
2. Add 2 polynomials
3. Exit
Enter choice: 1

Polynomial1:
Enter no. of terms: 3
Enter coeff: 2.0
Enter x exponent: 2
Enter y exponent: 0
Enter z exponent: 0

Enter coeff: -1.5
Enter x exponent: 1
Enter y exponent: 1
Enter z exponent: 0

Enter coeff: 3.5
Enter x exponent: 0
Enter y exponent: 3
```

```
Enter z exponent: 1

+2.0x^2y^0z^0-1.5x^1y^1z^0+3.5x^0y^3z^1
Enter x, y, and z values:
2
1
3

Sum = 14.0

1. Represent and evaluate polynomial
2. Add 2 polynomials
3. Exit
Enter choice: 2

Polynomial1:
Enter no. of terms: 2
Enter coeff: 4.0
Enter x exponent: 2
Enter y exponent: 1
Enter z exponent: 0

Enter coeff: 3.0
Enter x exponent: 0
Enter y exponent: 2
Enter z exponent: 1

Polynomial2:
Enter no. of terms: 3
Enter coeff: -2.0
Enter x exponent: 2
Enter y exponent: 1
Enter z exponent: 0

Enter coeff: 1.0
Enter x exponent: 1
Enter y exponent: 0
Enter z exponent: 1

Enter coeff: 2.5
Enter x exponent: 0
```

Enter y exponent: 3
Enter z exponent: 2

P1+P2:
+4.0x^2y^1z^0+3.0x^0y^2z^1+-2.0x^2y^1z^0+1.0x^1y^0z^1+2.5x^0y^3z^2
1. Represent and evaluate polynomial
2. Add 2 polynomials
3. Exit
Enter choice: 3

Program 10 :

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

```
#include<stdio.h>
#include<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("\nThe Postorder display : ");
                                postorder(root);
                            }

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

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

Output:

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

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

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

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

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

Enter Element to be searched: 66
~~~~

Key element is not found in the BST

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

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

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

Program11:

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

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

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

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

void dfs(int v)
{
        int i;
        visited[v]=1;
        s[++top] = v;
        for(i=1;i<=n;i++)
       {
              if(a[v][i] == 1&& visited[i] == 0 )
              {
```

```
                printf("%d ", i);
                dfs(i);
            }
        }
}

int main()
{

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

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

    printf("\n==>1. BFS: Print all nodes reachable from a given starting node");
    printf("\n==>2. DFS: Print all nodes reachable from a given starting node");
    printf("\n==>3:Exit");
    printf("\nEnter your choice: ");
    scanf("%d", &ch);
    switch(ch)
    {
      case 1: printf("\nNodes reachable from starting vertex %d are: ", start);
            bfs(start);
            for(i=1;i<=n;i++)
            {
              if(visited[i]==0)
                printf("\nThe vertex that is not reachable is %d" ,i);
            }
            break;
      case 2: printf("\nNodes reachable from starting vertex %d are:\n",start);
            dfs(start);
            break;
```

```
        case 3: exit(0);
        default: printf("\nPlease enter valid choice:");
      }
}
```

Output:

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

~~~Menu~~~~
==>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**
Enter the starting vertex: 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
~~~Menu~~~~
==>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**
Enter the starting vertex: 2
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
~~~Menu~~~~
==>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**
Enter the starting vertex: 1
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
~~~Menu~~~~
==>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**
Enter the starting vertex: 2
Nodes reachable from starting vertex 2 are: 3 4

Program12:

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

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 5
#define mod(x) x%MAX
void linear_prob(int a[],int num,int key)
{
   if(a[key]==-1)
      a[key]=num;
   else
   {
      printf("\nCollision detected!!");
      int i;
      for(i=mod(key+1);i!=key;i=mod(++i))
         if(a[i]==-1)
         break;
      if(i!=key)
      {
         printf("\nCollision avoided successfully\n");
         a[i]=num;
      }
      else
         printf("\nHash table is full\n");
   }
}
void display(int a[])
{
   short ch,i;
   printf("\n1.Filtered display\n2.Display all\nEnter choice:");
   scanf("%d",&ch);
   printf("\nHash table is :\n");
   for(i=0;i<MAX;i++)
      if(a[i]>0||ch-1)
         printf("%d  %d\n",i,a[i]);
```

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

}
```

Output:

Collision handling by linear probing

Enter the data: 7
Do you wish to continue (1/0): 1

Enter the data: 12
Collision avoided successfully
Do you wish to continue (1/0): 1

Enter the data: 7
Collision detected!!
Collision avoided successfully
Do you wish to continue (1/0): 1

Enter the data: 8
Collision avoided successfully
Do you wish to continue (1/0): 1

Enter the data: 18
Collision avoided successfully
Do you wish to continue (1/0): 1

Enter the data: 3
Collision detected!!
Collision avoided successfully
Do you wish to continue (1/0): 1

Enter the data: 14
Hash table is full
Do you wish to continue (1/0): 0

1.Filtered display
2.Display all
Enter choice:1

Hash table is :
1 18
2 8
3 3
4 14