ಬಿ.ಎಂ.ಎಸ್. ತಾಂತ್ರಿಕ ಮತ್ತು ವ್ಯವಸ್ಥಾಪನಾ ಮಹಾವಿದ್ಯಾಲಯ
(ವಿ.ಟಿ.ಯು. ಅಡಿಯಲ್ಲಿನ ಸ್ವಾಯತ್ತ ಸಂಸ್ಥೆ)
BMS INSTITUTE OF TECHNOLOGY & MANAGEMENT
(Autonomous Under VTU)

DATA STRUCTURES LABORATORY

MANUAL

Course Code: BCSL305

(Academic Year: 2024-25)

Department of CSE/ISE/AIML/CSBS
BMS Institute of Technology and Management
Bengaluru-560064

B.E. COMPUTER SCIENCE AND ENGINEERING

Choice Based Credit System (CBCS)

SEMESTER - III

DATA STRUCTURES LABORATORY (2:0:0) 1

(Effective from the academic year 2022-23)

Course Code	BCSL305	CIE Marks	50
Teaching Hours/Week (L:T:P)	0:0:2	SEE Marks	50
Total Number of Contact Hours	26	Exam Hours	02

Course Objectives:

This course enables students to:

1. Develop linear data structures and their applications such as stacks, queues and lists.
2. Develop non-linear data structures and their applications such as trees and graphs sorting and searching algorithms.

Descriptions:

Descriptions: Design, develop, and implement the specified Data Structure as given in the list given below using C Language under LINUX /Windows environment.

Sl. No	Programs List
1	Design, Develop and Implement a menu driven Program in C for the following operations on STACK of Integers (Array Implementation of Stack with maximum size MAX) a. Push an Element on to Stack b. Pop an Element from Stack c. Demonstrate Overflow and Underflow situations on Stack d. Display the status of Stack e. Exit Support the program with appropriate functions for each of the above operations.
2	Design, Develop and Implement a Program in C for converting an Infix Expression to Postfix Expression. Program should support for both parenthesized and free parenthesized expressions with the operators: +, -, *, /, %(Remainder), ^ (Power) and alphanumeric operands.
3	Design, Develop and Implement a Program in C for evaluation of Suffix expression with single digit operands and operators: +, -, *, /, %, ^.
4	Design, Develop and Implement a menu driven Program in C for the following operations on Circular QUEUE of integers (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.

5	Design, Develop and Implement a menu driven Program in C for the following operations on Double Ended QUEUE of integers (Array Implementation of Queue with maximum size MAX) a. Perform Insertion / Deletion at front of QUEUE b. Perform Insertion / Deletion at rear of QUEUE c. Display the status of Circular QUEUE d. Exit Support the program with appropriate functions for each of the above operations.
6	Design, Develop and Implement a menu driven Program in C for the following operations on Singly Linked List (SLL) of Student Data with the fields: USN, Name, Branch, Sem, PhNo a. Create a SLL of N Students Data by using front insertion. b. Display the status of SLL and count the number of nodes in it c. Perform Insertion / Deletion at End of SLL d. Perform Insertion / Deletion at Front of SLL(Demonstration of stack) e. Exit
7	Design, Develop and Implement a menu driven Program in C for the following operations on Doubly Linked List (DLL) of Employee Data with the fields: SSN, Name, Dept, Designation, Sal, PhNo a. Create a DLL of N Employees Data by using end insertion. b. Display the status of DLL and count the number of nodes in it c. Perform Insertion and Deletion at End of DLL d. Perform Insertion and Deletion at Front of DLL e. Demonstrate how this DLL can be used as Double Ended Queue f. Exit.
8	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
9	Design, Develop and Implement a Program in C for the following operations on Graph(G) of Cities a. Create a Graph of N cities using Adjacency Matrix. b. Print all the nodes reachable from a given starting node in a digraph using any traversal method (DFS/BFS).
10	Given a set of N employee records with a set K of Keys (4-digit) which uniquely determine the records. Assume that the records are available in the 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.

Course Outcomes: The student should be able to: **CO1:** Write programs to implement different types of data structures. **CO2:** Demonstrate the working of different types of data structures	
Textbooks	
1.	Ellis Horowitz and Sartaj Sahni, Fundamentals of Data Structures in C, Universities Press, 2nd edition, 2019
2.	P Reema Thareja, Data Structures using C, 3 rd Ed, Oxford press, 2012.

DSA Lab Evaluation Scheme

1. Ten marks for every experiment (10 X 10 = 100 marks), round it off to ***30 marks***.
2. Ten marks for every experiment will be evaluated for write-up, program execution, the procedure followed while execution and viva voce after each exercise.
3. Internal practical test for 100 marks to be given and the marks scored will be scaled down to ***20 marks***.
4. A Minimum of ***20 mark*** is to be scored in CIE.
5. SEE examination for the Lab is to be conducted for 100 marks and reduced to ***50 marks***.
6. A Minimum of ***18 marks*** is to be scored in SEE.

Note: Open Ended experiment will be done by the students in the Lab session. A total mark of 40 is to be scored by the student from both CIE and SEE together out of 100.

Exp:1 - Design, Develop and Implement a menu driven Program in C for the following Operations on STACK of Integers (Array Implementation of Stack with maximum size MAX)

a. Push an Element on to Stack
b. Pop an Element from Stack
c. Demonstrate Overflow and Underflow situations on Stack
d. Display the status of Stack
e. Exit

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

```
#include<stdio.h>
#include<stdlib.h>
#define SIZE 20
void push(int ele, int *top, int stack[]);
void pop(int *top, int stack[]);
void display(int top, int stack[]);
void main()
{
	int choice, top=-1, ele, flag;
	int stack[SIZE];
	for(;;)
	{
		printf("Enter\n1. Push\n2. Pop\n3. Display\n4. Exit\n");
		scanf("%d", &choice);
		switch(choice)
		{
			case 1: if(top==(SIZE-1))
					printf("Stack overflow!!!\n");
				else
				{
					printf("Enter element to be pushed:\n");
					scanf("%d", &ele);
					push(ele, &top, stack);
				}
				break;
			case 2: if(top==-1)
					printf("Stack underflow!!!\n");
				else
					pop(&top, stack);
				break;
			case 3:  if(top==-1)
					printf("Stack underflow!!!\n");
```

```
                                else
                                        display(top, stack);
                                break;
                        case 4: exit(0);
                }
        }
}
void push(int ele, int *top, int stack[])
{
        *top+=1;
        stack[*top]= ele;
}
void pop(int *top, int stack[])
{
        printf("Element to be deleted:\n%d\n", stack[*top]);
        *top-=1;
}
void display(int top, int stack[])
{
        int i;
        printf("Elements are:\n");
        for(i=top; i>=0; i--)
                printf("%d\t", stack[i]);
        printf("\n");
}
```

Exp:2 - Design, Develop and Implement a Program in C for converting an Infix Expression to Postfix Expression. Program should support for both parenthesized and free parenthesized expressions with the operators: +, -, *, /, %(Remainder), ^ (Power) and alphanumeric operands.

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

int G(char);
int F(char);
void infix_postfix(char infix[], char postfix[]);

int F(char sym)
{
	switch(sym)
	{
		case '+':
		case '-': return 2;
		case '*':
		case '/': return 4;
		case '^':
		case '$': return 5;
		case '(': return 0;
		case '#': return -1;
		default: return 8;
	}
}
int G(char sym)
{
	switch(sym)
	{
		case '+':
		case '-': return 1;
		case '*':
		case '/': return 3;
		case '^':
		case '$': return 6;
		case '(': return 9;
		case ')': return 0;
		default: return 7;
```

```
        }
}
void infix_postfix(char infix[], char postfix[])
{
        int top=-1,i,j=0;
        char stack[20],sym;
        stack[++top]='#';
        for(i=0; i<strlen(infix);i++)
        {
                sym = infix[i];
                while(F(stack[top]) > G(sym))
                        postfix[j++] = stack[top--];
                if(F(stack[top]) != G(sym))
                        stack[++top] = sym;
                else
                        top--;
        }
        while(stack[top] != '#')
                postfix[j++] = stack[top--];
        postfix[j] = '\0';
        return;
}

void main()
{
        char postfix[20],infix[20];
        printf("Enter the infix expression: ");
        scanf("%s",infix);
        infix_postfix(infix,postfix);
        printf("The Postfix expression is: ");
        printf("%s",postfix);
 }
```

Exp:3 - Design, Develop and Implement a Program in C for evaluation of Suffix expression with single digit operands and operators: +, -, *, /, %, ^.

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

double compute(double op1, double op2, char sym)
{
   switch (sym)
   {
      case '+' : return (op1+op2);
      case '-' : return (op1-op2);
      case '*' : return (op1*op2);
      case '/' : return (op1/op2);
      case '$' :
      case '^' : return (pow(op1,op2));
   }
}

void main()
{
   int i, top=-1;
   char postfix[20], sym;
   double s[20],op1,op2,res;
   printf("\nEnter postfix expression : ");
   scanf("%s",postfix);
   for(i=0; i<strlen(postfix); i++)
   {
      sym=postfix[i];
      if(isdigit(sym))
         s[++top]=sym-'0';
      else
      {
         op2=s[top--];
         op1=s[top--];
         res=compute(op1,op2,sym);
         s[++top]=res;
      }
```

```
    }
    res=s[top--];
    printf("\nThe result of the expression is : %.4f\n",res);
    return;
}
```

Exp:4 - Design, Develop and Implement a menu driven Program in C for the following operations on Circular QUEUE of integers (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 QSIZE 5
int cq_full(int count)
{
   return((count == QSIZE) ? 1 : 0);
}
int cq_empty(int count)
{
   return((count == 0) ? 1 : 0);
}
void cq_insert(int q[], int* r, int* count)
{
   int item;
   if (cq_full(*count))
   {
      printf("\n Queue is full \n");
      return;
   }
   printf("\n Enter item to be inserted:");
   scanf("%d", &item);
   *r = (*r + 1) % QSIZE;
   q[*r] = item;
   (*count)++;
   return;
}
void cq_delete(int q[], int* f, int* count)
{
   if (cq_empty(*count))
   {
      printf("\n Queue is empty \n");
```

```
        return;
    }
    printf("\n Element deleted is %d\n", q[*f]);
    *f = (*f + 1) % QSIZE;
    (*count)--;
    return;
}
void display(int q[], int f, int count)
{
    if (cq_empty(count))
    {
        printf("\n Queue is empty \n");
        return;
    }
    printf("\n Queue elements:");
    int j = f, i;
    for (i = 0; i < count; i++)
    {
        printf("%d\t", q[j]);
        j = (j + 1) % QSIZE;
    }
    return;
}
void main()
{
    int choice, f = 0, r = -1, count = 0;
    int q[QSIZE];
    while (1)
    {
        printf("\n Enter the choice: \n1. INSERT\n 2.DELETE\n 3.DISPLAY\n 4.EXIT\n");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1: cq_insert(q, &r, &count);
            break;
        case 2: cq_delete(q, &f, &count);
            break;
        case 3: display(q, f, count);
            break;
        case 4:exit(0);
        default: printf("\n Invalid choice");
        }
    } }
```

Exp:5 - Design, Develop and Implement a menu driven Program in C for the following operations on Double Ended QUEUE of integers (Array Implementation of Queue with maximum size MAX)

a. Perform Insertion / Deletion at front of QUEUE
b. Perform Insertion / Deletion at rear of QUEUE
c. Display the status of Circular QUEUE
d. Exit

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

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

int queue_full(int rear)
{
   return ((rear==SIZE-1)?1:0);
}

int queue_empty(int rear,int front)
{
   return((rear<front)?1:0);
}

void queue_insert_rear(int queue[],int *rear,int ele)
{
   if(queue_full(*rear))
   {
      printf("\n queue full");
      return;
   }
   queue[++(*rear)]=ele;
}

void queue_insert_front(int queue[],int *rear,int *front,int ele)
{
   if(queue_empty(*rear,*front))
   {
      queue[++(*rear)]=ele;
      return;
   }
   if (*front!=0)
   {
```

```
        queue[--(*front)]=ele;
        return;
    }
}

void queue_delete_rear(int queue[],int *rear,int *front)
{
    if(queue_empty(*rear,*front))
    {
        printf("\n queue empty");
        return;
    }
    printf("the element deleted is %d",queue[*rear]);
    if(*front>*rear)
    {
        *front=0;
        *rear=-1;
        return;
    }
    else
    {
        *rear=(*rear-1+SIZE)%SIZE;
        return;
    }
}

void queue_delete_front(int queue[],int *rear,int *front)
{
    if (queue_empty(*rear, *front)) {
        printf("\n queue empty");
        return;
    }
    printf("the element deleted is %d", queue[*front]);
    if (*front > *rear) {
        *front = 0;
        *rear = -1;
        return;
    } else {
        *front = (*front + 1) % SIZE;
        return;
    }
}
```

```
void display(int queue[],int rear,int front)
{
   int i;
   if (queue_empty(rear,front))
   {
      printf("\n queue empty");
      return;
   }
   printf("the element in the dequeue are:");
   for(i=front;i<=rear;i++)
   {
      printf("%d\t",queue[i]);
   }
}

int  main()
{
   int Queue[SIZE],rear=-1,front=0,choice,item;

   for(;;)
   {
      printf("\n1.Insert Front \n2.Insert Rear \n3.Delete Front \n4.Delete Rear
\n5.Display \n6.Exit");
      printf("\nEnter choice:");
      scanf("%d",&choice);
      switch(choice)
      {
         case 1 : printf("Enter the element :");
            scanf("%d",&item);
            queue_insert_front(Queue,&rear,&front,item);
            break;
         case 2 : printf("Enter the element :");
            scanf("%d",&item);
            queue_insert_rear(Queue,&rear, item);
            break;
         case 3 :
            queue_delete_front(Queue,&rear,&front);
            break;
         case 4 :
            queue_delete_rear(Queue,&rear,&front);
            break;
         case 5: display(Queue,rear,front);
            break;
```

```
            case 6 : exit(0);
            default : printf("enter valid choice");

              }
          }
      return 0;
}
```

Exp:6 - Design, Develop and Implement a menu driven Program in C for the following operations on Singly Linked List (SLL) of Student Data with the fields: USN, Name, Branch, Sem, PhNo

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

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
struct node
{
        char name[20];
        char usn[10];
        char branch[5];
        int sem;
        char phno[10];
        struct node* link;
};
typedef struct node* NODE;
//get node
NODE get_node()
{
        NODE temp;
        temp = (NODE)malloc(sizeof(struct node));
        temp->link=NULL;
        printf("\n Name: ");
        scanf("%s", temp->name);
        printf("\n USN: ");
        scanf("%s", temp->usn);
        printf("\n Branch: ");
        scanf("%s", temp->branch);
        printf("\n Sem: ");
        scanf("%d", &temp->sem);
        printf("\n Ph no: ");
        scanf("%s", temp->phno);
        return temp;
}
//insert front
NODE insert_front(NODE first)
```

```
{
        NODE temp;
        temp = get_node();
        if(first==NULL)
           return temp;
   else
   {
           temp->link = first;
           return temp;
   }
}
//insert rear
NODE insert_rear(NODE first)
{
        NODE temp, next;
        temp = get_node();
        if (first == NULL)
                return temp;
        else
        {
           next = first;
           while (next->link != NULL)
                   next = next->link;
           next->link = temp;
        }
        return first;
}
//delete front
NODE del_front(NODE first)
{
        NODE temp;
        if (first == NULL)
        {
                printf("\n No entries");
                return first;
        }
        temp = first;
        first=first->link;
        free(temp);
        return first;
}
//delete rear
NODE del_rear(NODE first)
```

```
{
        NODE temp, prev, cur;
        if (first == NULL)
        {
                printf("\n No entries \n");
                return first;
        }
        else
                if (first->link == NULL)
                {
                        printf(" Deleted USN:%s \n",first->usn);
                        free(first);
                        return first;
                }
        cur = first;
        while (cur->link != NULL)
        {
                prev = cur;
                cur = cur->link;
        }
        prev->link = NULL;
        printf("Deleted USN: %s",cur->usn);
        free(cur);
        return first;
}
//display
void display(NODE first)
{
        NODE temp = first;
   int count=0;
        if(temp==NULL)
        {
           printf("\n No entries \n");
           return;
        }
        printf("\n Student info: \n");
        while (temp != NULL)
        {

           printf("Name:%s\t", temp->name);
        printf("USN:%s\t", temp->usn);
           printf("Branch:%s\t", temp->branch);
           printf("Sem:%d\t", temp->sem);
```

```
        printf("Ph no:%s\t", temp->phno);
        printf("\n");
      count++;
            temp = temp->link;
        }
   printf("\n the no. of nodes are %d",count);
}
//main func
void main()
{
        NODE first = NULL;
        int choice, item;
        for (;;)
        {
            printf("\n\n Enter choice 1.IF 2.IR 3.DF 4.DR 5.display 6.Exit\n");
            scanf("%d", &choice);
            switch (choice)
            {
            case 1: first = insert_front(first);
                    break;
            case 2: first = insert_rear(first);
                    break;
            case 3: first = del_front(first);
                    break;
            case 4: first = del_rear(first);
                    break;
            case 5: display(first);
                    break;
            case 6: exit(0);
            }
        }
}
```

Exp:7 - Design, Develop and Implement a menu driven Program in C for the following operations on Doubly Linked List (DLL) of Employee Data with the fields: SSN, Name, Dept, Designation, Sal, PhNo

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

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

struct node
{
  int SSN;
  char Name[20];
  char Dept[20];
  char Designation[20];
  int Sal;
  char Ph_No[11];
  struct node *llink;
  struct node *rlink;
};
typedef struct node *NODE;

NODE get_node();
NODE insert_front(NODE first);
NODE insert_rear(NODE first);
NODE delete_front(NODE first);
NODE delete_rear(NODE first);
void DLL_display(NODE first);

void main()
{
  NODE first=NULL;
  int choice;
  for(;;)
  {
    printf("\nEnter\n1.Insert from front\n2.Insert from rear\n3.Delete from
front\n4.Delete from rear\n5.Display\n6.Exit:\n");
    scanf("%d",&choice);
```

```
      switch(choice)
      {
        case 1:first=insert_front(first);
               break;
        case 2:first=insert_rear(first);
               break;
        case 3:first=delete_front(first);
               break;
        case 4:first=delete_rear(first);
               break;
        case 5:DLL_display(first);
               break;
        case 6:exit(0);
        default:printf("\nWrong Choice!");
      }//End of switch case
    }//End of for loop
    return;
}//End of main()

NODE get_node()
{
    NODE temp;
    temp=(NODE)malloc(sizeof(struct node));
    temp->llink=temp->rlink=NULL;
    printf("\nEnter the details of Employee:\n");
    printf("Employee SSN:");
    scanf("%d",&temp->SSN);
    printf("Employee Name:");
    scanf("%s",temp->Name);
    printf("Employee Department:");
    scanf("%s",temp->Dept);
    printf("Employee Designation:");
    scanf("%s",temp->Designation);
    printf("Employee Salary:");
    scanf("%d",&temp->Sal);
    printf("Employee Phone Number:");
    scanf("%s",temp->Ph_No);
    return temp;
}
NODE insert_front(NODE first)
{
    NODE temp;
    temp=get_node();
```

```
  if(first==NULL)
    return temp;
  temp->rlink=first;
  first->llink=temp;
  return temp;
}
NODE insert_rear(NODE first)
{
  NODE temp,next;
  temp=get_node();
  if(first==NULL)
     return temp;
  next=first;
  while(next->rlink!=NULL)
    next=next->rlink;
  next->rlink=temp;
  temp->llink=next;
  return first;
}
NODE delete_front(NODE first)
{
  NODE temp;
  if(first==NULL)
  {
     printf("\nNo nodes in the DLL!");
     return first;//return NULL
  }
  temp=first;
  printf("\nThe node to be deleted is:%d\t%s\t%s\t%s\t%d\t%s",temp->SSN,temp-
>Name,temp->Dept,temp->Designation,temp->Sal,temp->Ph_No);
  first=first->rlink;
  first->llink=NULL;
  free(temp);
  return first;
}
NODE delete_rear(NODE first)
{
  NODE prev,cur;
  if(first==NULL)
  {
     printf("\nNo nodes in the DLL!");
     return first;//return NULL
  }
```

```
  if(first->rlink==NULL)
  {
     printf("\nThe node to be deleted is:%d\t%s\t%s\t%s\t%d\t%s",first->SSN,first-
>Name,first->Dept,first->Designation,first->Sal,first->Ph_No);
     free(first);
     return NULL;
  }//End of if block
  cur=first;
  while(cur->rlink!=NULL)
  {
     prev=cur;
     cur=cur->rlink;
  }//End of while loop
  printf("\nThe node to be deleted is:%d\t%s\t%s\t%s\t%d\t%s",cur->SSN,cur-
>Name,cur->Dept,cur->Designation,cur->Sal,cur->Ph_No);
  prev->rlink=NULL;
  free(cur);
  return first;
}//End of delete_rear()
void DLL_display(NODE first)
{
  NODE temp;
  int count=0;
  if(first==NULL)
  {
     printf("\nNumber of nodes in the DLL is: %d",count);
     printf("\nNo nodes in the DLL!");
     return;
  }
  temp=first;
  printf("\nThe information of Employee(s) are: ");
  while(temp!=NULL)
  {
    printf("\nEmployee SSN: %d",temp->SSN);
    printf("\nEmployee Name: %s",temp->Name);
    printf("\nEmployee Department: %s",temp->Dept);
    printf("\nEmployee Designation: %s",temp->Designation);
    printf("\nEmployee Salary: %d",temp->Sal);
    printf("\nEmployee Ph_No: %s\n",temp->Ph_No);
    count++;
    temp=temp->rlink;
  }//End of while loop
  printf("\nThe number of nodes in the DLL are: %d",count);
```

```
    return;
}//End of LL_display()
```

Exp:8 - 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 tnode
{
   int info;
   struct tnode* llink;
   struct tnode* rlink;

};

typedef struct tnode * TNODE;

TNODE get_node ()
{
   TNODE temp;
   temp = (TNODE)malloc(sizeof(struct tnode));
   temp->llink=temp->rlink=NULL;
   return temp;
}

TNODE insert (TNODE root, int ele)
{
   if (root == NULL)
   {
      TNODE temp;
      temp=get_node();
      temp->info=ele;
      return temp;
   }
   if (ele<root->info)
      root->llink=insert(root->llink,ele);
   if (ele>root->info)
      root->rlink=insert(root->rlink,ele);
```

```
    return root;
}

void preorder(TNODE root)
{
    if (root!=NULL)
    {
        printf("%d\t",root->info);
        preorder(root->llink);
        preorder(root->rlink);
    }
    return;
}

void postorder(TNODE root)
{
    if (root!=NULL)
    {
        postorder(root->llink);
        postorder(root->rlink);
        printf("%d\t",root->info);
    }
    return;
}

void inorder(TNODE root)
{
    if (root!=NULL)
    {
        inorder(root->llink);
        printf("%d\t",root->info);
        inorder(root->rlink);
    }
    return;
}

int search(TNODE root,int key)
{
    if (root!=NULL)
    {
        if (root->info==key)
            return key;
        if (key<root->info)
```

```
        return search(root->llink,key);
      return search(root->rlink,key);
   }
   return -1;
}

void main()
{
        TNODE root = NULL;
        int choice, ele, key;
        printf("\n1. Insert \n2. Search\n3. Display\n4. Exit\n");
        for (;;)
        {
                printf(" \nEnter your choice : ");
                scanf("%d", &choice);
                switch (choice)
                {
                case 1: printf("Enter the element : ");
                      scanf("%d",&ele);
                      root = insert(root,ele);
                            break;
                case 2: printf("Enter the element to be searched : ");
                      scanf("%d",&ele);
                      key = search(root,ele);
                      if (key==-1)
                         printf("Key not Found\n");
                      else
                         printf("Key Found\n");
                            break;
                case 3: printf("Preorder : ");
                      preorder(root);
                      printf("\nInorder : ");
                      inorder(root);
                      printf("\nPostorder : ");
                      postorder(root);
                      printf("\n");
                            break;
                case 4: exit(0);
                default : printf("Invalid Input\n");
                }
        }
}
```

Exp:9 - Design, Develop and Implement a Program in C for the following operations on Graph(G) of Cities

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

b. Print all the nodes reachable from a given starting node in a digraph using any traversal method (DFS/BFS).

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

int city[10][10],v[10],queue[10],r=0,f=1,n;

void dfs(int s)
{
   int i;
   v[s]=1;
   for (i=1;i<=n;i++)
         if (city[s][i] && !v[i])
         {
                  printf("%d\t",i);
                  v[i]=1;
                  dfs(i);
         }
   return;
}

void bfs(int s)
{
   int i;
   for (i=1;i<=n;i++)
         if (city[s][i] && !v[i])
         {
                  queue[++r]=i;
                  printf("%d\t",i);
         }
   if (f<=r)
   {
         v[queue[f]]=1;
         bfs(queue[++f]);
   }
   return;
}
```

```
void main()
{
   int count=0,i,choice,j,s;
   printf("\nEnter the number of cities : ");
   scanf("%d",&n);
   printf("\nEnter the adjacency matrix for connected cities : \n");
   for (i=1;i<=n;i++)
   {
         v[i]=0;
         queue[i]=0;
         for (j=1;j<=n;j++)
               scanf("%d",&city[i][j]);
   }
   printf("\nEnter 1 for dfs and 2 for bfs : ");
   scanf("%d",&choice);
   printf("\nEnter the city to check connectivity of : ");
   scanf("%d",&s);
   printf("\nCities reachable from %d are : \n",s);
   if (choice==1)
         dfs(s);
   if (choice==2)
   {
         queue[++r]=s;
         bfs(s);
   }
   printf("\n");
   for (i=1;i<=n;i++)
         count+=v[i];
   if (count==n)
         printf("\n All cities are connected \n\n");
   else
         printf("\nAll cities are not connected \n\n");
}
```

Exp:10 - Given a set of N employee records with a set K of Keys (4-digit) which uniquely determine the records. Assume that the records are available in the 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>
typedef struct employee {
        int id;
        char name[20];
        int sal;
}EMP;
EMP emp[100];
int a[100], empID[100], count = 0;
int getemp(EMP emp[], int key, int ID) {
        FILE* fp;
        fp = fopen("out.txt", "a+");
        emp[key].id = ID;
        printf("\nEnter employee name and salary:\n");
        scanf("%s%d", emp[key].name, &emp[key].sal);
        fprintf(fp, "\n%d\t%s\t%d", emp[key].id, emp[key].name, emp[key].sal);
        fclose(fp);
        return key;
}
void display() {
        int i;
        printf("\nKey\tID\tName\tSalary");
        for (i = 0; i < 100; i++)
                if (a[i] != -1)
                        printf("\n%d\t%d\t%s\t%d", i, emp[i].id, emp[i].name, emp[i].sal);
}
void probe(int key, int ID) {
        int i = key, flag = 0;
        if (count == 100) {
                printf("Hash table is full.\n");
                exit(0);
        }
        if (a[key] == -1) {
```

```
            a[key] = getemp(emp, key, ID);
            display();
            count++;
        }
        else {
            printf("\nCollision detected. Solving it with linear probing...\n");
            while (a[i] != -1) {
                i++;
                if (i == 100) {
                    i = 0; // 1st change made here
                    while (i < key && a[i] != -1) // 2nd change made here
                        i++;
                }
            }
            a[i] = getemp(emp, i, ID);
            printf("\nCollision problem solved! Hash table:\n");
            display();
            count++;
        }
}
void main() {
        int key, i, j = 0, ans = 1;
        for (i = 0; i < 100; i++)
            a[i] = -1;
        do {
            printf("\nEnter the employee ID: ");
            scanf("%d", &empID[j]);
            key = empID[j] % 100;
            probe(key, empID[j]);
            printf("\n\nDo you want to continue the input? (1=Yes | 0=No): ");
            scanf("%d", &ans);
            j++;
        } while (ans);
        display(emp);
        for (i = 0; i < 100; i++)
            if (a[i] != -1)
                printf(" \t%d", a[i]);
        printf("\n");
}
```

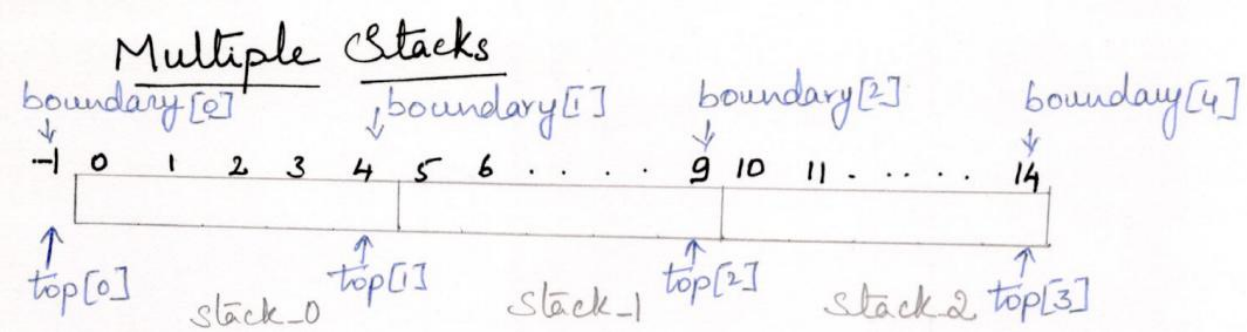
Open ended Program for DSA

2024-25

1. Design, Develop and Implement a multi stack Program in C. Design this program with menu options and let the multi-stack house minimum of three stacks in it. Incorporate the following functions in the program.

a. Push an Element on to a specific Stack
b. Pop an Element from a specific Stack
d. Demonstrate Overflow and Underflow situations in each stack
e. Display the status of Stack/Multi-Stack
f. Exit

Support the program with conditions for each of the above operations



2. Design, Develop and Implement a menu driven Program in C for the following operations on Circular Singly Linked List (SLL) of Doctors Data with the fields: SSN, Name, Specialization, Experience, PhNo.

a. Display the status of circular SLL and count the number of nodes in it.
b. Perform Insertion / Deletion at End of circular SLL.
c. Perform Insertion / Deletion at Front of circular SLL.
d. Search for a doctor with a specialization.
e. Exit

3. Design, Develop and Implement Floyd's algorithm in C. Accept the graph representing the distance between cities(as cost matrices) and find the shortest path from a city to all other cities in the graph.