NAGARJUNA COLLEGE OF ENGINEERING AND TECHNOLOGY
Venkatagiri kote post, Devanahalli, Bengaluru – 562164
(An Autonomous College under VTU, Accredited with NAAC "A+")

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Data Structure and Applications

22CSI32

LIST OF LABORATORY PROGRAMS			
Course Code	**22CSI32**	**CIE Marks**	**50**
Teaching Hours/Week (L: T: P: S) (3:0:2:0)	**Credits (3:0:0:0)**	**SEE Marks**	**50**
Total Hours of Pedagogy	**40 Hours Theory 10 Lab slots**	**Total Marks**	**100**
Credits	**03**	**Exam Hours**	**03**

Course objectives:

This course will enable students to experience practically on:

- Understand the basics of pointers and dynamics memory allocation.
- Learn concepts of structure and its applications in programming.
- Gain knowledge to choose the specific linked lists for implementing real world problems.
- Grasp the use of stacks and queues in solving complex problems.
- Acquire knowledge of nonlinear data structure like trees.

List of the problems for which the students has to develop the programs and execute in the laboratory

1. Design, develop and execute a program in C based on the following requirements: An EMPLOYEE structure is to contain the following members: Employee_Number (an integer), Employee_Name (a string of characters), Basic_Salary (an integer), All_Allowances (an integer), IT (an integer), Net_Salary (an integer). Write a functions to read the data of an employee, to calculate Net_Salary and to print the values of all the structure members. (All_Allowances = 123% of Basic, Income Tax (IT) = 30% of the gross salary (gross salary = Basic_Salary + All_Allowance), Net_Salary = Basic_Salary + All_Allowances –IT).

2. Write a program to Store Roll number of N students. Perform Insert and delete Roll_No at a given valid position (POS) using pointers. Display the status of array elements at any given point of time. Support the program with functions for each operations.

3. Develop an array implementation on stack and perform Push and Pop operations. Check for overflow and underflow conditions. Demonstrate stack implementation to check palindrome. Display the status of the stack for all the operations performed. Support the program with appropriate functions for each of the above operations.

4. Write a Program in C for converting an Infix Expression to Postfix Expression. Program should support for both parenthesized and free parenthesized expressions with arithmetic operators. Evaluate the Suffix (Postfix) expression with single digit operands and operators.

5. Implement Circular QUEUE program in C for rainbow colors (VIBGOYR) and perform Insert and Delete operations. Check for overflow and underflow conditions. Display the status of the Circular QUEUE for all the operations performed. Use pointers and functions.

6. Implement a Menu driven Program in C for the following operations on Singly Linked List (SLL) of Student Data with the fields: *USN, Name, Avg_Marks*

a) Create N number of Students Data by inserting at end of the list.
b) Insert and Delete at front of the list
c) Delete at the end of list
d) Display the status of SLL
e) Demonstration stack and queue
f) Exit

7. Design and Develop following operations on Doubly Linked List (DLL) of Employee Data with the fields: *SSN, Name, Dept etc.*

a) Create a Node of N Employees Data by inserting in front.
b) Insert a new node to the right of key value.
c) Perform Insertion and Deletion at End of DLL
d) Display the status of DLL and count the number of nodes
e) Exit

10. Design and Develop a program in C for the following operations on Binary Search Tree (BST) of Integers.

a. Create a BST of N Integers
b. Traverse the BST using Inorder, Preorder and Post Order techniques
c. Search a KEY element in BST and display the appropriate message

Steps to Execute C Program in Windows OS

Step-1:Write a C Program
Open turbo C Editor or Dev C++,write C program and save the program called source code.

Step-2:compile the program(Alt+F9)
Compiler is used to compile the program to check errors and also convert high level code to machine level code.

Step-3:RUN (Ctrl+F9)
Final step is to run C program to see output. (by pressing ctrl+9)

1. Design, develop and execute a program in C based on the following requirements: An EMPLOYEE structure is to contain the following members: Employee_Number (an integer), Employee_Name (a string of characters), Basic_Salary (an integer), All_Allowances (an integer), IT (an integer), Net_Salary (an integer). Write a functions to read the data of an employee, to calculate Net_Salary and to print the values of all the structure members. (All_Allowances = 123% of Basic, Income Tax (IT) = 30% of the gross salary (gross salary = Basic_Salary + All_Allowance), Net_Salary = Basic_Salary + All_Allowances –IT).

```
#include<stdio.h>
int i,n;
struct EMP
{
  char emp_name[20];
  int emp_no, basic;
  float gross, net_salary, all_allowances, IT;

}e[5];

void Readdata()
{
        printf("enter how many emps?");
        scanf("%d", &n);

        printf("\nemployee name\t   EMP Num\t  Basic sal \n");
        for (i=0;i<n; i++)
                scanf("%s%d%d",e[i].emp_name,&e[i].emp_no,&e[i].basic);

}

void calculate()
{
        for (i=0;i<n; i++)
        {
                e[i].all_allowances=1.23*e[i].basic;
                e[i].gross=e[i].basic+e[i].all_allowances;
                e[i].IT=0.3*e[i].gross;
                e[i].net_salary=e[i].gross-e[i].IT;

         }

}

void Displaydata()
{
         printf("Employee Details are : \n");
         printf("Emp Name\t Emp Num\t Basic\t gross\t net_salary\t all_allowances\t IT\n ");
         for (i=0;i<n; i++)
        printf("%s\t %d\t %d\t %f\t %f\t
%f\t%f\n",e[i].emp_name,e[i].emp_no,e[i].basic,e[i].gross, e[i].net_salare[i].all_allowances, e[i].IT);
}
```

```
int main()
{
        Readdata();
        calculate();
        Displaydata();
}
```

2. Write a program to Store Roll number of N students. Perform Insert and delete Roll_No at a given valid position (POS) using pointers. Display the status of array elements at any given point of time. Support the program with functions for each operations.

```
#include<stdio.h>
#include<stdlib.h>
int a[20],n;
void create()
{
        int i;
        printf("\n Enter n");
        scanf("%d",&n);
        printf("\n Enter the elements");
        for(i=0;i<n;i++)
        scanf("%d",&a[i]);
}
void display()
{
        int i;
        printf("\n The elements are");
        for(i=0;i<n;i++)
        printf("%d",a[i]);
}
void insert()
{
        int i,key,pos;
        printf("\n Enter key and position");
        scanf("%d%d",&key,&pos);
        for(i=n-1;i>=pos-1;i--)
         a[i+1]=a[i];
        a[i+1]=key;
      n++;
}
void del()
{
        int i,pos;
        printf("\n Enter the position to be deleted");
        scanf("%d",&pos);
        for(i=pos;i<n;i++)
        a[i-1]=a[i];
        n--;
}
int main()
{
        int c,a;
        while(1)
        {
                printf("\n Enter 1 to create, Enter 2 to display, Enter 3 to insert, Enter 4 to delete");
                scanf("%d",&c);
                switch(c)
                {
```

```
                case 1:printf("\n Enter n");
                scanf("%d",&n);
                create(&n);break;
                case 2:display();break;
                case 3:insert();break;
                case 4:del();break;
                default:exit(0);break;
              }
        }
}
```

3. Develop an array implementation on stack and perform Push and Pop operations. Check for overflow and underflow conditions. Demonstrate stack implementation to check palindrome. Display the status of the stack for all the operations performed. Support the program with appropriate functions for each of the above operations

```
#include<stdio.h>
#include<stdlib.h>
#define max 4
int a[10],top=-1;

void push(int m)
{
          if(top==max-1)
           printf("\n Stack overflow");
  else
           a[++top]=m;
}

int pop()
{
          if(top==-1)
               printf("\n Stack underflow");
  else
return a[top--];
}

void display()
{
int i;
    if(top==-1)
         printf("\n Stack is empty");
   else
       {
printf("\n The elements are");
for(i=top;i>=0;i--)
printf("%d\t",a[i]);
         }
}
void palin()
{
int n,num,rem=0,flag=1;
printf("\n Enter n");
scanf("%d",&n);
num=n;
  while(n!=0)
        {
     rem=n%10;
               push(rem);
               n=n/10;
        }
 while(num!=0)
```

```
    {
if(num%10!=pop())
{
        flag=0;
        break;
}
num=n/10;
    }
if(flag==0)
printf("\n It is not a palindrome");
else
printf("\n It is a palindrome");
}

int main()
{
int c,m;
while(1)
{
printf("\n 1 push\n 2 pop\n 3 display\n 4 palindrome");
 scanf("%d",&c);
 switch(c)
 {
          case 1: printf("\n Enter an element\t");
           scanf("%d",&m);
           push(m);
           break;
           case 2: printf("\n The popped elementis %d", pop());
           break;
           case 3: display();break;
           case 4:palin();break;
           default:exit(0);break;
}
 }
return 0;
 }
```

4. Write a Program in C for converting an Infix Expression to Postfix Expression. Program should support for both parenthesized and free parenthesized expressions with arithmetic operators. Evaluate the Suffix (Postfix) expression with single digit operands and operators.

```
#include<stdio.h>
#include<string.h>
 char stack[20];
 int top=-1;

void push(char s)
{
  stack[++top]=s;
}

char pop()
{
 return stack[top--];
}

int precd(char s)
{
    switch(s)
            {
                 case '^': return 4;
                case '*':case '/': return 3;
                case '+': case '-': return 2;
                 case '(': case ')': case '#':return 1;
             }
 }

void convertip(char infix[], char postfix[])
{
int i,j=0;
char symb;
push('#');
for(i=0;i<strlen(infix);i++)
{
symb=infix[i];
switch(symb)
{
case '(': push(symb);break;
case ')': while(stack[top]!='(')
postfix[j++]=pop();
pop();
break;
case '^': case '*': case '/': case '+': case '-':
while(precd(symb)<=precd(stack[top]))
postfix[j++]=pop();
push(symb);
break;
```

```
                    default: postfix[j++]=symb;
}
}
while(stack[top]!='#')
 postfix[j++]=pop();
}

int main()
{
char infix[20],postfix[20];
printf("\n Enter infix expression");
gets(infix);
convertip(infix,postfix);
puts(postfix);
return 0;
}
```

5. Implement Circular QUEUE program in C for rainbow colors (VIBGOYR) and perform Insert and Delete operations. Check for overflow and underflow conditions. Display the status of the Circular QUEUE for all the operations performed. Use pointers and functions.

```
#include<stdio.h>
#include<stdlib.h>
#define size 7
char cq[size];
int r=-1,f=0,cnt=0;

void cq_insert()
{
        if(cnt==size)
        {
                printf("Cq is full\n");
                return;
        }
        printf("Enter the Rainbow color\n");
        r=(r+1)%size;
        scanf("%s",&cq[r]);

        cnt++;
}

void del()
{
        if(cnt==0)
        {
                printf("Cq is empty\n");
                return;
        }
        printf("The deleted Rainbow color is %c",cq[f]);
        f=(f+1)%size;
        cnt--;
}

void cq_display()
{
        int i,k=f;
        if(cnt==0)
        {
                printf("Cq is empty\n");
                return;
        }
        printf("\n Cq Rainbow color are\n");
        for(i=0;i<cnt;i++)
        {
                printf("%c",cq[k]);
                k=(k+1)%size;
        }
}
```

```
int main()
{
        int ch;
        while(1)
        {
                printf("\n1.Cq_insert\n 2.cq_delete\n 3.cq_display\n 4.exit\n");
                printf("Enter the choice\n");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:cq_insert();break;
                        case 2:del();break;
                        case 3:cq_display();break;
                        case 4:exit(0);break;
                        default:printf("Invalid input");
                }
        }
        return 0;
}
```

6. Implement a Menu driven Program in C for the following operations on Singly Linked List (SLL) of Student Data with the fields: USN, Name, Avg_Marks

- ***a) Create N number of Students Data by inserting at end of the list.***
- ***b) Insert and Delete at front of the list***
- ***c) Delete at the end of list***
- ***d) Display the status of SLL***
- ***e) Demonstration stack and queue***
- f) ***Exit***

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
struct sll
{
        char usn[10];
        struct sll *next;
};
typedef struct sll node;
node *start=NULL;
node *getnode()
{
        node *n1;
        n1=(node*)malloc(sizeof(node));
        n1->next=NULL;
        printf("\n enter the usn\n");
        scanf("%s",&n1->usn);
        return n1;
}
void insert_front()
{
        node *n1;
        n1=getnode();
        if(start==NULL)
        {
                start=n1;
                return;
        }
        n1->next=start;
        start=n1;
}
void insert_end()
{
        node *n1,*temp=start;
        n1=getnode();
        if(start==NULL)
        {
                start=n1;
                return;
        }
        while(temp->next!=NULL)
        temp=temp->next;
```

```
        temp->next=n1;
}
void delete_front()
{
        node *temp=start;
        if(start==NULL)
        {
                printf("\n the list is empty\n");
                return;
        }
        start=start->next;
        printf("\nthe deleted usn is %s\n",temp->usn);
        free(temp);
}
void delete_end()
{
        node *temp=start,*prev;
        if(start==NULL)
        {
                printf("\n the list is empty\n");
                return;
        }
        if(start->next==NULL)
        {
                start=NULL;
                        printf("\nthe deleted usn is %s\n",temp->usn);
        free(temp);
        return;
        }
        while(temp->next!=NULL)
        {
                prev=temp;
                temp=temp->next;
        }
        prev->next=NULL;
        printf("\nthe deleted usn is %s\n",temp->usn);
        free(temp);
}
void display()
{
        node *temp=start;
        int cnt=0;
        if(start==NULL)
        {
                printf("\n the list is empty\n");
                return;
        }
        printf("\n the elements in the list are\n");
        while(temp!=NULL)
        {
                printf("%s\n",temp->usn);
```

```
                temp=temp->next;
                cnt++;
        }
        printf("\n total number of node in the list are %d\n",cnt);
}
void main()
{
        int i,ch,n;
        while(1)
        {
                printf("\nenter the choice\n");
                printf("1.insert front\t2.insert end\t3.delete front\t4.delete end\t5.display\n");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:printf("enter the n value\n");
                            scanf("%d",&n);
                            for(i=0;i<n;i++)
                            {
                                insert_front();
                            }
                                break;
                        case 2:insert_end();break;
                        case 3:delete_front();break;
                        case 4:delete_end();break;
                        case 5:display();break;
                        default:printf("\n enter the correct choice\n");
                        break;
                }
        }
}
```

7. Design and Develop following operations on Doubly Linked List (DLL) of Employee Data with the fields: SSN, Name, Dept etc.

- ***a) Create a Node of N Employees Data by inserting in front.***
- ***b) Insert a new node to the right of key value.***
- ***c) Perform Insertion and Deletion at End of DLL***
- ***d) Display the status of DLL and count the number of nodes***
- **e) *Exit***

```
#include<stdio.h>
#include<stdlib.h>
struct dll
{
        int info;
        struct dll *rptr,*lptr;
};
typedef struct dll node;
node *start=NULL;
node *getnode_dll()
{
        node *p;
        p=(node*)malloc(sizeof(node));
        p->lptr=p->rptr=NULL;
        printf("\n enter the info\n");
        scanf("%d",&p->info);
        return p;
}
void insert_front()
{
        node *n1;
        n1=getnode_dll();
        if(start==NULL)
        {
                start=n1;
                return;
        }
        n1->rptr=start;
        start->lptr=n1;
        start=n1;
}
void insert_end()
{
        node *n1,*temp=start;
        n1=getnode_dll();
        if(start==NULL)
        {
                start=n1;
                return;
        }
        while(temp->rptr!=NULL)
        temp=temp->rptr;
        temp->rptr=n1;
```

```
        n1->lptr=temp;
}
void delete_front()
{
        node *temp=start;
        if(start==NULL)
        {
                printf("\n the list is empty\n");
                return;
        }
        start=start->rptr;
        start->lptr=NULL;
        printf("\n the deleted element is %d",temp->info);
        free(temp);
}
void delete_end()
{
        node *temp=start;
                if(start==NULL)
        {
                printf("\n the list is empty\n");
                return;
        }
        if(start->rptr==NULL)
        {
                start=NULL;
                printf("\n the deleted element is %d\n",temp->info);
                free(temp);
                return;
        }
        while(temp->rptr!=NULL)
        temp=temp->rptr;
        (temp->lptr)->rptr=NULL;
        printf("\n the deleted element is %d",temp->info);
        free(temp);
}
void display()
{
        node *temp=start;
        int cnt=0;
                if(start==NULL)
        {
                printf("\n the list is empty\n");
                return;
        }
        printf("\n the elements in the list are:\n");
        while(temp!=NULL)
        {
                printf("%d\t",temp->info);
                temp=temp->rptr;
                cnt++;
```

```
        }
        printf("\n the no of nodes in list are :%d",cnt);
}
void main()
{
        int ch;
        while(1)
        {
                printf("\nenter the choice\n");
                printf("\n1.insert front\t2.insert end\t3.delete begin\t4.delete end\t5.display\n");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:insert_front();break;
                        case 2:insert_end();break;
                        case 3:delete_front();break;
                        case 4:delete_end();break;
                        case 5:display();break;
                        default:printf("enter the correct choice");
                        break;
                }
        }
}
```

8. Design and Develop a program in C for the following operations on Binary Search Tree (BST) of Integers.

a) Create a BST of N Integers
b) Traverse the BST using Inorder, Preorder and Post Order techniques
c) Search a KEY element in BST and display the appropriate message

```
#include<stdio.h>
#include<stdlib.h>
struct BST
{
        int info;
        struct BST *lptr,*rptr;
};
typedef struct BST node;
node *root=NULL;
void inorder(node *root)
{
        if(root!=NULL)
        {
                inorder(root->lptr);
                printf("%d\n",root->info);
                inorder(root->rptr);
  }
}
void preorder(node *root)
{
        if(root!=NULL)
        {
                printf("%d\n",root->info);
                preorder(root->lptr);
                preorder(root->rptr);
  }
}
void postorder(node *root)
{
        if(root!=NULL)
        {
                postorder(root->lptr);
                postorder(root->rptr);
                printf("%d\n",root->info);
  }
}
node *insert(node *root,int key)
{
        if(root==NULL)
        {
                root=(node*)malloc(sizeof(node));
                root->lptr=root->rptr=NULL;
                root->info=key;
                return root;
```

```
        }
        if(key<root->info)
        {
                root->lptr=insert(root->lptr,key);
        }
        else if(key>root->info)
        {
           root->rptr=insert(root->rptr,key);
  }
        else if(key==root->info)
        {
                printf("\nNO DUPLICATES ARE ALLOWED");
        }
        return root;
}
int search(node *root,int key)
{
        while(root!=NULL)
        {
                if(key<root->info)
                {
                        return search(root->lptr,key);
                }
                else if(key>root->info)
                {
                         return search(root->rptr,key);
                }
                else
                  return 1;
  }
        return 0;

}
void main()
{
        int i,n,key,ch,flag;
        while(1)
        {
                printf("\n enter the choice\n");
                printf("1.insert\t2.search\t3.inorder\t4.preorder\t5.postoder\n");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:printf("\n enter the n value\n");
                             scanf("%d",&n);
                             printf("\n enter the %d key values\n",n);
                             for(i=0;i<n;i++)
                             {
                                  scanf("%d",&key);
                                  root=insert(root,key);
                                    }
```

```
                    break;
                case 2:printf("\n enter the key value\n");
                    scanf("%d",&key);
                    flag=search(root,key);
                    if(flag==1)
                    printf("ele found\n");
                    else
                    printf("ele not found\n");
                    break;
                case 3:inorder(root);break;
                case 4:preorder(root);break;
                case 5:postorder(root);break;
                default:printf("enter the correct choice\n");break;

            }
        }
}
```