ADICHUNCHANAGIRI INSTITUTE OF TECHNOLOGY

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

Data Structures Lab Manual BCSL305

2023-24

Odd Semester

Prepared By,

Ashwini Kamath

Asst. Professor

IS&E Department

Lab Program List

ADICHUNCHANAGIRI INSTITUTE OF TECHNOLOGY

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

Institution Vision and Mission

Vision

To develop Adichunchanagiri Institute of Technology as a center of excellence and to strive for continuous improvement of technical education and human resource advancement.

Mission

To achieve Excellence in Education, Entrepreneurship, and Innovation by producing Engineers with high Ethical Standards, Integrity, and Credibility.

ADICHUNCHANAGIRI INSTITUTE OF TECHNOLOGY

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

Department Vision and Mission

Vision

To be recognized as a center of excellence in information technology and allied areas with quality learning and research environment

Mission

- Provide intellectual & professional leadership in ethical and social areas pertaining to information in contemporary society.
- Advancing the state of knowledge of information studies through research and development.
- Providing a platform to discuss cutting edge technologies.

ADICHUNCHANAGIRI INSTITUTE OF TECHNOLOGY

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

Subject: Data Structures Laboratory
Subject code: BCSL305
Semester: III

Course Outcomes

At the end of the course the student will be able to:

CO1	Analyze various linear and non-linear data structures.
CO2	Demonstrate the working nature of different types of data structures and their applications
CO3	Use appropriate searching and sorting algorithms for the give scenario.
CO4	Apply the appropriate data structure for solving real world problems

1. Design, Develop and Implement a menu driven Program in C for the following Array operations
a. Creating an Array of N Integer Elements
b. Display of Array Elements with Suitable Headings
c. Inserting an Element (ELEM) at a given valid Position (POS)
d. Deleting an Element at a given valid Position(POS)
e. Exit.
Support the program with functions for each of the above operations.

```
#include<stdio.h>
#include<stdlib.h>
int a[20];
int n,val,i,pos;
/*Function Prototype*/
void create();
void display();
void insert();
void delete();
int main()
{
        int ch;
        clrscr();
        do
        {
                printf("--------MENU-----------\n");
                printf("1.CREATE\n");
                printf("2.DISPLAY\n");
                printf("3.INSERT\n");
                printf("4.DELETE\n");
                printf("5.EXIT\n");
                printf("----------------------\n");
                printf("Enter your choice\n");
                scanf("%d",&ch);
                switch(ch)
                {
```

```
                        case 1: create();
                                break;
                        case 2: display();
                                break;
                        case 3: insert();
                                break;
                        case 4: delete();
                                break;
                        case 5: exit(0);
                                break;
                        default:printf("Invalid choice:\n");
                                break;
                }
        }       while(ch<6);
        return 0;
}
/*creating an array*/
void create()
{
        printf("Enter the size of the array elements:\n");
        scanf("%d",&n);
        printf("Enter the elements for the array:\n");
        for(i=0;i<n;i++)
                scanf("%d",&a[i]);

}
/*displaying an array elements*/
void display()
{
        int i;
        printf("The array elements are:\n");
```

```
        for(i=0;i<n;i++)
                printf("%d\t",a[i]);
        printf("\n");

}
/*inserting an element into an array*/
void insert()
{
        printf("Enter the position for the new element\n");
        scanf("%d",&pos);
        printf("Enter the element to be inserted\n");
        scanf("%d",&val);
        for(i=n-1;i>=pos;i--)
                a[i+1]=a[i];

        a[pos]=val;
        n=n+1;
}
/*deleting an array element*/
void delete()
{
        printf("Enter the position of the element to be deleted:\n");
        scanf("%d",&pos);
        val=a[pos];
        for(i=pos;i<n-1;i++)
                a[i]=a[i+1];

        n=n-1;
        printf("The deleted element is =%d\n",val);
}
```

Output:

```
--------MENU------
1.CREATE
2.DISPLAY
3.INSERT
4.DELETE
5.EXIT
------------------
Enter your choice
1
Enter the size of the array elements:
5
Enter the elements for the array:
10 20 30 40 50
--------MENU-----------
1.CREATE
2.DISPLAY
3.INSERT
4.DELETE
5.EXIT
----------------------
Enter your choice
2
The array elements are:
10      20      30      40      50
--------MENU-----------
1.CREATE
2.DISPLAY
3.INSERT
4.DELETE
5.EXIT
----------------------
Enter your choice
3
Enter the position for the new element
2
Enter the element to be inserted
1000
--------MENU-----------
1.CREATE
2.DISPLAY
3.INSERT
4.DELETE
5.EXIT
----------------------
Enter your choice
```

```
2
The array elements are:
10      20      1000    30      40      50
--------MENU-----------
1.CREATE
2.DISPLAY
3.INSERT
4.DELETE
5.EXIT
----------------------
Enter your choice
4
Enter the position of the element to be deleted:

3
The deleted element is =30
--------MENU-----------
1.CREATE
2.DISPLAY
3.INSERT
4.DELETE
5.EXIT
----------------------
Enter your choice
2
The array elements are:
10      20      1000    40      50
--------MENU-----------
1.CREATE
2.DISPLAY
3.INSERT
4.DELETE
5.EXIT
----------------------
Enter your choice
5
```

2. Design, Develop and Implement a Program in C for the following operations on Strings

a. Read a main String (STR), a Pattern String (PAT) and a Replace String (REP)

b. Perform Pattern Matching Operation: Find and Replace all occurrences of PAT in STR with REP if PAT exists in STR. Report suitable messages in case PAT does not exist in STR. Support the program with functions for each of the above operations. Don't use Built-in functions.

```
#include<stdio.h>
int main()
{
   char STR[20],PAT[20],REP[20],ans[30];
   int i=0,j=0,k=0,n=0,flag;
   printf("Enter the MAIN string\n");
   gets(STR);
   printf("Enter the PATTERN string\n");
   gets(PAT);
   printf("Enter the REPLACE string\n");
   gets(REP);
   for(i=0;STR[i]!='\0';i++)
   {
        flag=1;
        for(j=0;PAT[j]!='\0';j++)
                if(STR[i+j]!=PAT[j])
                        flag=0;
        n=j;
        if(flag)
        {
```

```
                        for(j=0;REP[j]!='\0';j++,k++)
                        ans[k]=REP[j];
                        i+=n-1;
        }
        else
                        ans[k++]=STR[i];
    }
    ans[k]='\0';
    for(i=0;ans[i]!='\0';i++)
            STR[i]=ans[i];
    STR[i]='\0';
    printf("The RESULTANT string is:%s\n" ,STR);
    return 0;
}
```

Output:

```
Enter the MAIN string
AIT CKM
Enter a PATTERN string
CKM
Enter a REPLACE string
Information Science
The RESULTANT string is: AIT Information Science
```

3. Design, Develop and Implement a menu driven Program in C for the following operations on STACK of Integers (Array Implementation of Stack with maximum size MAX)
a. *Push* an Element on to Stack
b. *Pop* an Element from Stack
c.* Demonstrate how Stack can be used to check *Palindrome
d. Demonstrate *Overflow* and *Underflow* situations on Stack
e. Display the status of Stack
f. Exit
Support the program with appropriate functions for each of the above operations

```
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
#define max 5
int stack[max],top=-1;
void push();
void pop();
void display();
void pal();
int main()
{
        int choice;
        while(choice)
        {
                printf("\n\n--------STACK OPERATIONS-----------\n");
                printf("1.Push\n");
                printf("2.Pop\n");
                printf("3.Palindrome\n");
                printf("4.Display\n");
                printf("5.Exit\n");
                printf("-----------------------");
                printf("\nEnter your choice:\t");
                scanf("%d",&choice);
                switch(choice)
                {
```

```
                        case 1: push();
                                break;
                        case 2: pop();
                                break;
                        case 3: pal();
                                break;
                        case 4: display();
                                break;
                        case 5: exit(0);
                                 break;
                        default:printf("\nInvalid choice:\n");
                                break;
                }
        }
        return 0;
}
void push() //Inserting element into the stack
{
        int item,n;
        if(top==(max-1))
                printf("\nStack Overflow:");
        else
        {
                printf("Enter the element to be inserted:\t");
                scanf("%d",&item);
                stack[++top]=item;
        }
}
void pop() //deleting an element from the stack
{
        if(top==-1)
```

```
                printf("Stack Underflow:");
        else
                printf("\nThe poped element: %d\t",stack[top--]);
}
void pal()
{
        int j,k,len=top+1,flag=0,i=0,length=0;
        int num[len],rev[len];
        while(top!=-1)
                num[i++]= stack[top--];
        for(j=0;j<len;j++)
                printf("Numbers= %d\n",num[j]);
        i=0;
        printf("reverse operation : \n");
        for(k=len-1;k>=0;k--)
                rev[k]=num[i++];
        printf("reverse array : ");
        for(k=0;k<len;k++)
                printf("%d\n",rev[k]);
        printf("check for palindrome :\n");
        for(i=0;i<len;i++)
                if(num[i]==rev[i])
                        length = length+1;
        if(length==len)
                printf("It is palindrome number\n");
        else
                printf("It is not a palindrome number\n");
        top = len-1;
}
```

```
void display()
{
        int i;
        if(top==-1)
                printf("\nStack is Empty:");
        else
                printf("\nThe stack elements are:\n" );
        for(i=top;i>=0;i--)
                printf("%d\n",stack[i]);
}
```

Output:

```
--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
----------------------
Enter your choice:      1
Enter the element to be inserted:       1

--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
----------------------
Enter your choice:      1
Enter the element to be inserted:       2

--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
----------------------
Enter your choice:      1
Enter the element to be inserted:       1

--------STACK OPERATIONS-----------
```

```
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
----------------------
Enter your choice:      3
Numbers= 1
Numbers= 2
Numbers= 1
reverse operation :
reverse array : 1
2
1
check for palindrome :
It is palindrome number

--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
----------------------
Enter your choice:      1
Enter the element to be inserted:       2

--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
----------------------
Enter your choice:      3
Numbers= 2
Numbers= 1
Numbers= 2
Numbers= 1
reverse operation :
reverse array : 1
2
1
2
check for palindrome :
It is not a palindrome number
```

```
--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
----------------------
Enter your choice:      3
Numbers= 2
Numbers= 1
Numbers= 2
Numbers= 1
reverse operation :
reverse array : 1
2
1
2
check for palindrome :
It is not a palindrome number
--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
----------------------
Enter your choice:      2

The poped element: 2

--------STACK OPERATIONS-----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
----------------------
Enter your choice:      4

The stack elements are:
1
2
1
```

```
--------STACK OPERATIONS----------
1.Push
2.Pop
3.Palindrome
4.Display
5.Exit
----------------------
Enter your choice:      5
```

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

```
#include<stdio.h>
#define MAX 100
typedef enum{lparen,rparen,plus,minus,mult,divide,mod,pow,eos,operand}precedence;
char stack[MAX];
int top=-1;
precedence gettoken(char*,int*);
void push(char);
char pop();
void stackfull();
void stackempty();
void postfix();
void printtoken(char);
char infix[100];
int main()
{
        printf("enter an infix expression\n");
        scanf("%s",infix);
        printf("postfix form\n");
        postfix();
        return 0;
}
int isp[]={0,19,12,12,13,13,13,14,0};
int icp[]={20,19,12,12,13,13,13,14,0};
void postfix()
{
        char symbol;
        precedence token;
```

```
        int n=0;
        for(token=gettoken(&symbol,&n);token!=eos;token=gettoken(&symbol,&n))
        {
                if(token==operand)
                        printf("%c",symbol);
                else if(token==rparen)
                {
                        while(stack[top]!=lparen)
                                printtoken(pop());
                        pop();
                }
                else
                {
                        while(isp[stack[top]]>=icp[token])
                                printtoken(pop());
                        push(token);
                }
        }
        while(top!=-1)
                printtoken(pop());
        printf("\n");
}
precedence gettoken(char *symbol,int *n)
{
        *symbol=infix[(*n)++];
        switch(*symbol)
        {
                case '(':return lparen;
                case ')':return rparen;
                case '+':return plus;
                case '-':return minus;
```

```
                case '*':return mult;
                case '/':return divide;
                case '%':return mod;
                case '^':return power;
                case '\0':return eos;
                default :return operand;
        }
}
void printtoken(char token)
{
        switch(token)
        {
                case plus:printf("+");
                        break;
                case minus:printf("-");
                        break;
                case mult:printf("*");
                        break;
                case divide:printf("/");
                        break;
                case mod:printf("%%");
                        break;
                case pow:printf("^");
                        break;
        }
}
void push(char item)
{
        if(top==MAX-1)
                stackfull();
        stack[++top]=item;
```

```
}
char pop()
{
        if(top==-1)
                stackempty();
        return stack[top--];
}
void stackfull()
{
        fprintf(stderr,"stack is full\n");
        return;
};
void stackempty()
{
        fprintf(stderr,"stack is empty\n");
        return;
}
```

Output:

```
enter an infix expression
1+2*3^4-(5%3)/2
postfix form
1234^*+53%2/-
```

5. Design, Develop and Implement a Program in C for the following Stack Applications
a. Evaluation of Suffix expression with single digit operands and operators: +, -, *, /, %, ^
b. Solving Tower of Hanoi problem with n disks

```
#include<stdio.h>
#include<math.h>
#define MAX 100
typedef enum{plus,minus,mult,mod,div,power,eos,operand}precedence;
void push(int);
int top=-1;
int pop();
int stackfull();
int eval();
precedence gettoken(char *,int *);
int stack[MAX];
char postfix[MAX];
int main()
{
        int result;
        printf("enter postfix expression\n");
        scanf("%s",postfix);
                result=eval();
        printf("postfix evaluation result=%d\n",result);
        return 0;
}
int eval()
{
        precedence token;
        char symbol;
        int op1,op2;
        int n=0;
        int top=-1;
        token=gettoken(&symbol,&n);
```

```
        while(token!=eos)
        {
                if(token==operand)
                        push(symbol-'0');
                else
                {
                        op2=pop();
                        op1=pop();
                        switch(token)
                        {
                                case plus:push(op1+op2);
                                        break;
                                case minus:push(op1-op2);
                                        break;
                                case mult:push(op1*op2);
                                        break;
                                case div:push(op1/op2);
                                        break;
                                case mod:push(op1%op2);
                                        break;
                                case power:push(pow(op1,op2));
                                        break;
                        }
                }
                token=gettoken(&symbol,&n);
        }
        return pop();
}
precedence gettoken(char *symbol,int *n)
{
        *symbol=postfix[(*n)++];
```

```
        switch(*symbol)
        {
                case '+':return plus;
                case '-':return minus;
                case '*':return mult;
                case '/':return div;
                case '%':return mod;
                case '^':return power;
                case '\0':return eos;
                default:return operand;
        }
}
void push(int element)
{
        if(top==MAX-1)
        stackfull();
        stack[++top]=element;
}
int stackfull()
{
        fprintf(stderr,"stack is full\n");
        return;
}
int pop()
{
        if(top==-1)
        stackempty();
        return stack[top--];
}
int stackempty()
{
```

```
        fprintf(stderr,"stack is empty\n");
        return;
}
```

Output:

```
[root@localhost Desktop]# cc eval.c -lm
[root@localhost Desktop]# ./a.out
enter postfix expression
12+
postfix evaluation result=3
[root@localhost Desktop]# ./a.out
enter postfix expression
1234^*2/+12%-
postfix evaluation result=81
[root@localhost Desktop]# ./a.out
enter postfix expression
23^
postfix evaluation result=8
```

Tower Hanoi

```
#include<stdio.h>
void tower(int,char,char,char);
int main()
{
  int n;
  printf("enter the number of disk\n");
  scanf("%d",&n);
  printf("the sequence of disk induced in power of hanoi are\n");
  tower(n,'A','C','B');
   return 0;
 }
void tower(int n,char from,char to,char aux)
{
  if(n==1)
  {
    printf("move disk 1 from peg %c to peg %c\n",from,to);
    return;
   }
    tower(n-1,from,aux,to);
    printf("move disk %d from peg %c to peg %c\n",n,from,to);
    tower(n-1,aux,to,from);
 }
```

Output:

```
enter the number of disk
3
the sequence of disk induced in power of hanoi are
move disk 1 from peg A to peg C
move disk 2 from peg A to peg B
move disk 1 from peg C to peg B
move disk 3 from peg A to peg C
move disk 1 from peg B to peg A
move disk 2 from peg B to peg C
move disk 1 from peg A to peg C
```

6. Design, Develop and Implement a menu driven Program in C for the following operations on Circular QUEUE of Characters (Array Implementation of Queue with maximum size MAX)
a. Insert an Element on to Circular QUEUE
b. Delete an Element from Circular QUEUE
c. Demonstrate *Overflow* and *Underflow* situations on Circular QUEUE
d. Display the status of Circular QUEUE
e. Exit
Support the program with appropriate functions for each of the above operations

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 5
char q[MAX];
int frount=MAX-1,rear=MAX-1;
void insert();
void delet();
void display();
int main()
{
        int ch;
        printf("circular queue operation\n");
        while(1)
        {
                printf("1.insert\n2.delete\n3.display\n4.exit\n");
                printf("enter your choice\n");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:insert();
                                break;
                        case 2:delet();
                                break;
                        case 3:display();
                                break;
```

```
                        case 4:exit(1);
                                break;
                        default:printf("invalid choice\n");
                                break;
                }
        }
        return 0;
}
void insert()
{
        char item;
        printf("enter the element to be added\n");
        scanf(" %c",&item);
        if(frount==((rear+1)%MAX))
                printf("queue is overflow\n");
        else
        {
                rear=(rear+1)%MAX;
                q[rear]=item;
        }
}
void delet()
{
        if(frount==rear)
                printf("queue is underflow\n");
        else
        {
                frount=(frount+1)%MAX;
                printf("deleted element=%c\n",q[frount]);
        }
}
```

```
void display()
{
	int i;
	if(frount==rear)
		printf("queue is underflow\n");
	else
	{
		if(frount>rear)
		{
			for(i=frount+1;i<=MAX-1;i++)
				printf("%c\t",q[i]);
			for(i=0;i<=rear;i++)
				printf("%c\t",q[i]);
		}
		else
			for(i=frount+1;i<=rear;i++)
				printf("%c\t",q[i]);
		printf("\n");
	}
}
```

Output:

```
circular queue operation
1.insert
2.delete
3.display
4.exit
enter your choice
2
queue is underflow
1.insert
2.delete
3.display
4.exit
enter your choice
1
```

```
enter the element to be added
a
1.insert
2.delete
3.display
4.exit
enter your choice
1
enter the element to be added
b
1.insert
2.delete
3.display
4.exit
enter your choice
1
enter the element to be added
c
1.insert
2.delete
3.display
4.exit
enter your choice
1
enter the element to be added
d
1.insert
2.delete
3.display
4.exit
enter your choice
1
enter the element to be added
e
queue is overflow
1.insert
2.delete
3.display
4.exit
enter your choice
2
deleted element=a
1.insert
2.delete
3.display
4.exit
enter your choice
```

```
2
deleted element=b
1.insert
2.delete
3.display
4.exit
enter your choice
1
enter the element to be added
a
1.insert
2.delete
3.display
4.exit
enter your choice
3
c       d       a
1.insert
2.delete
3.display
4.exit
enter your choice
4
```

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

a. Create a SLL of N Students Data by using *front insertion*.
b. Display the status of SLL and count the number of nodes in it
c. Perform Insertion and Deletion at End of SLL
d. Perform Insertion and Deletion at Front of SLL
e. Demonstrate how this SLL can be used as STACK and QUEUE
f. Exit

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int count=0;
struct node
{
        int sem;
        char name[20],branch[20],usn[20],phno[20];
        struct node *next;
};
struct node *first=NULL,*last=NULL,*temp=NULL,*templ;
void create();
void insertfirst();
void insertlast();
void display();
void deleteend();
void deletefirst();
void main()
{
        int ch,n,i;
        first =NULL;
        temp=templ=NULL;
        printf("MENU\n");
        printf("1.create\n");
```

```
        printf("2.display\n3.insert at end\n4.delete at end\n5.insert at begining\n6.delete at
begining\n7.exit\n");
        while(1)
        {
                printf("enter your choice\n");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:printf("enter number of student\n");
                                scanf("%d",&n);
                                for(i=0;i<n;i++)
                                        insertfirst();
                                        break;
                        case 2:display();
                                break;
                        case 3:insertlast();
                                break;
                        case 4:deleteend();
                                break;
                        case 5:insertfirst();
                                break;
                        case 6:deletefirst();
                                break;
                        case 7:exit(0);
                        default:printf("invalid choice\n");
                                break;
                }
        }
}
void create()
{
```

```
        int sem;
        char name[20],branch[20],usn[20],phno[20];
        temp=(struct node*)malloc(sizeof(struct node));
        printf("enter usn,name,branch,sem and phone no of student\n");
        scanf("%s%s%s%d%s",usn,name,branch,&sem,phno);
        strcpy(temp->usn,usn);
        strcpy(temp->name,name);
        strcpy(temp->branch,branch);
        strcpy(temp->phno,phno);
        temp->sem=sem;
        count++;
}
void insertfirst()
{
        if(first==NULL)
        {
                create();
                first=temp;
                last=temp;
        }
        else
        {
                create();
                temp->next=first;
                first=temp;
        }
}
void insertlast()
{
        if(first==NULL)
        {
```

```
                create();
                first=temp;
                last=temp;
        }
        else
        {
                create();
                last->next=temp;
                last=temp;
        }
}
void display()
{
        templ=first;
        if(first==NULL)
                printf("list empty\n");
        else
        {
                printf("content of list\n");
                printf("usn\tname\tbranch\tsem\tphno\n");
                for(templ=first;templ!=NULL;templ=templ->next)
                        printf("%s\t%s\t%s\t%d\t%s\n",templ->usn,templ->name,templ-
>branch,templ->sem,templ->phno);
                printf("number of nodes in list=%d\n",count);
        }
}
void deleteend()
{
        temp=first;
        if(temp->next==NULL)
        {
```

```
                free(temp);
                first=NULL;
                last=NULL;
        }
        else
        {
                for(temp=first;temp->next!=last;temp=temp->next)
                ;
                printf("deleted node\n");
                printf("%s\t%s\t%s\t%d\t%s\n",last->usn,last->name,last->branch,last->sem,last-
>phno);
                free(last);
                temp->next=NULL;
                last=temp;
                count--;
        }
}
void deletefirst()
{
        temp=first;
        if(first==NULL)
                printf("list is empty\n");
        else
        {
                printf("deleted node is\n");
                if(temp->next==NULL)
                {
                        printf("%S\t%s\t%s\t%d\t%s\n",temp->usn,temp->name,temp-
>branch,temp->sem,temp->phno);
                        free(temp);
                        first=last=NULL;
```

```
                }
                else
                {
                        printf("%s\t%s\t%s\t%d\t%s\n",temp->usn,temp->name,temp-
>branch,temp->sem,temp->phno);
                        first=temp->next;
                        free(temp);
                }
                count--;
        }
}
```

Output:

```
MENU
1.create
2.display
3.insert at end
4.delete at end
5.insert at begining
6.delete at begining
7.exit
enter your choice
2
list empty
enter your choice
1
enter number of student
2
enter usn,name,branch,sem and phone no of student
4AI15IS002 bbb ISE 3 9854326754
enter usn,name,branch,sem and phone no of student
4AI15IS001 aaa ISE 3 9976543265
enter your choice
2
content of list
usn     name   branch sem     phno
4AI15IS001    aaa     ISE     3       9976543265
4AI15IS002    bbb     ISE     3       9854326754
number of nodes in list=2
enter your choice
3
enter usn,name,branch,sem and phone no of student
```

```
4AI15IS003 ccc ISE 3 8796541234
enter your choice
5
enter usn,name,branch,sem and phone no of student
4AI14IS031 ggg ISE 3 9534267512
enter your choice
2
content of list
usn      name   branch sem     phno
4AI14IS031    ggg    ISE    3       9534267512
4AI15IS001    aaa    ISE    3       9976543265
4AI15IS002    bbb    ISE    3       9854326754
4AI15IS003    ccc    ISE    3       8796541234
number of nodes in list=4
enter your choice
4
deleted node
4AI15IS003    ccc    ISE    3       8796541234
enter your choice
6
deleted node is
4AI14IS031    ggg    ISE    3       9534267512
enter your choice
2
content of list
usn      name   branch sem     phno
4AI15IS001    aaa    ISE    3       9976543265
4AI15IS002    bbb    ISE    3       9854326754
number of nodes in list=2
enter your choice
7
```

8. Design, Develop and Implement a menu driven Program in C for the following operations on Doubly Linked List (DLL) of Employee Data with the fields: *SSN, Name, Dept, Designation, Sal, PhNo*
a. Create a DLL of N Employees Data by using *end insertion*.
b. Display the status of DLL and count the number of nodes in it
c. Perform Insertion and Deletion at End of DLL
d. Perform Insertion and Deletion at Front of DLL
e. Demonstrate how this DLL can be used as Double Ended Queue
f. Exit

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int count=0;
struct listnode
{
        struct listnode *prev;
        char ssn[20],phno[20];
        float sal;
        char name[20],dept[20],desg[20];
        struct listnode *next;
};
typedef struct listnode node;
node *first,*last,*temp;
void create();
void insertbeg();
void insertend();
void display();
void deleteend();
void deletebeg();
int main()
{
        int ch,n,i;
        first=NULL;
        temp=NULL;
```

```
        last=NULL;
        printf("MENU\n");
        printf("1.create\n2.display\n3.insert end\n4.delete end\n5.insert beg\n6.delete
beg\n7.exit\n");
        while(1)
        {
                printf("enter your choice\n");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:printf("enter the number of emplloyes\n");
                                scanf("%d",&n);
                                for(i=0;i<n;i++)
                                        insertend();
                                break;
                        case 2:display();
                                break;
                        case 3:insertend();
                                break;
                        case 4:deleteend();
                                break;
                        case 5:insertbeg();
                                break;
                        case 6:deletebeg();
                                break;
                        case 7:exit(0);
                                break;
                        default:printf("invalid choice\n");
                                break;
                }
        }
```

```
        return 0;
}
void create()
{
        char ssn[20],phno[20],name[20],dept[20],desg[20];
        float sal;
        temp=(node*)malloc(sizeof(node));
        temp->prev=temp->next=NULL;
        printf("enter ssn,name,department,designation,salary and ph.no of emplloyee\n");
        scanf("%s%s%s%s%f%s",ssn,name,dept,desg,&sal,phno);
        strcpy(temp->ssn,ssn);
        strcpy(temp->name,name);
        strcpy(temp->dept,dept);
        strcpy(temp->desg,desg);
        strcpy(temp->phno,phno);
        temp->sal=sal;
        count++;
}
void insertbeg()
{
        if(first==NULL)
        {
                create();
                first=last=temp;
        }
        else
        {
                create();
                temp->next=first;
                first->prev=temp;
                first=temp;
```

```
        }
}
void insertend()
{
        if(first==NULL)
        {
                create();
                first=last=temp;
        }
        else
        {
                create();
                temp->prev=last;
                last->next=temp;
                last=temp;
        }
}
void deleteend()
{
        if(first==NULL)
        {
                printf("list is empty\n");
                return;
        }
        else if(first->next==NULL)
        {
                temp=first;
                printf("deleted node contains\n");
                printf("%s\t%s\t%s\t%s\t%f\t%s\n",temp->ssn,temp->name,temp->dept,temp-
>desg,temp->sal,temp->phno);
                free(temp);
```

```
                first=last=NULL;
                return;
        }
        else
        {
                temp=last;
                last=last->prev;
                printf("deleted node contains\n");
                printf("%s\t%s\t%s\t%s\t%f\t%s\n",temp->ssn,temp->name,temp->dept,temp-
>desg,temp->sal,temp->phno);
                free(temp);
                last->next=NULL;
        }
}
void deletebeg()
{       if(first==NULL)
        {
                printf("list is empty\n");
                return;
        }
        else if(first->next==NULL)
        {
                temp=first;
                printf("deleted node contains\n");
                printf("%s\t%s\t%s\t%s\t%f\t%s\n",temp->ssn,temp->name,temp->dept,temp-
>desg,temp->sal,temp->phno);
                free(temp);
                first=last=NULL;
                return;
        }
        else
```

```
        {
                temp=first;
                first=first->next;
                printf("deleted node contains\n");
                printf("%s\t%s\t%s\t%s\t%f\t%s\n",temp->ssn,temp->name,temp->dept,temp-
>desg,temp->sal,temp->phno);
                free(temp);
                first->prev=NULL;
        }
}
void display()
{
        if(first==NULL)
        {
                printf("list is empty\n");
                return;
        }
        else
        {
                printf("ssn\tname\tdept\tdesg\tsal\tphno\n");
                for(temp=first;temp!=NULL;temp=temp->next)
                        printf("%s\t%s\t%s\t%s\t%f\t%s\n",temp->ssn,temp->name,temp-
>dept,temp->desg,temp->sal,temp->phno);
        }
}
```

Output:

```
MENU
1.create
2.display
3.insert end
4.delete end
5.insert beg
6.delete beg
7.Exit
```

```
enter your choice
2
list is empty
enter your choice
1
enter the number of emplloyes
2
enter ssn,name,department,designation,salary and ph.no of emplloyee
1001 aaa ISE ASST.Prof 20000 8765412378
enter ssn,name,department,designation,salary and ph.no of emplloyee
1002 bbb ISE ASST.Prof 20000 9856432512
enter your choice
2
ssn     name   dept    desg    sal     phno
1001    aaa    ISE     ASST.Prof     20000.000000 8765412378
1002    bbb    ISE     ASST.Prof     20000.000000 9856432512
enter your choice
3
enter ssn,name,department,designation,salary and ph.no of emplloyee
1003 ccc CSE ASST.Prof 20000 9856341234
enter your choice
5
enter ssn,name,department,designation,salary and ph.no of emplloyee
1000 ddd ISE ASST.Prof 25000 9678541234
enter your choice
2
ssn     name   dept    desg    sal     phno
1000    ddd    ISE     ASST.Prof     25000.000000 9678541234
1001    aaa    ISE     ASST.Prof     20000.000000 8765412378
1002    bbb    ISE     ASST.Prof     20000.000000 9856432512
1003    ccc    CSE     ASST.Prof     20000.000000 9856341234
enter your choice
4
deleted node contains
1003    ccc    CSE     ASST.Prof     20000.000000 9856341234
enter your choice
6
deleted node contains
1000    ddd    ISE     ASST.Prof     25000.000000 9678541234
enter your choice
2
ssn     name   dept    desg    sal     phno
1001    aaa    ISE     ASST.Prof     20000.000000 8765412378
1002    bbb    ISE     ASST.Prof     20000.000000 9856432512
enter your choice
7
```

9. Design, Develop and Implement a Program in C for the following operations on Singly Circular Linked List (SCLL) with header nodes
a. Represent and Evaluate a Polynomial P(x,y,z) = 6x2y2z-4yz5+3x3yz+2xy5z-2xyz3
b. Find the sum of two polynomials POLY1(x,y,z) and POLY2(x,y,z) and store the result in POLYSUM(x,y,z)
Support the program with appropriate functions for each of the above operations

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
struct listnode
{
        int coef;
        int ex,ey,ez;
        struct listnode *link;
};
typedef struct listnode node;
node *a,*b,*c,*d;
int co, expox, expoy, expoz, x, y, z;
node *attach(node *p,int co,int expox,int expoy,int expoz);
int eval();
int m, n;
node *padd();
void display(node *p);
int compare(int x1,int x2,int y1,int y2,int z1,int z2);
int main()
{
        int i, j, ch, res;
        a=b=c=NULL;
        while(1)
        {
                printf("Polynomial operations\n");
                printf("1.Evaluation\n2.Addition of two polynomials\n3.Exit\n");
                printf("Enter your choice\n");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:printf("Enter the no. of terms in polynomial\n");
                                scanf("%d",&n);
                                for(i=0;i<n;i++)
                                {
                                        printf("Enter coefficient,exponent of x,y,z\n");
                                        scanf("%d%d%d%d",&co,&expox,&expoy,&expoz);
                                        d=attach(d,co,expox,expoy,expoz);
                                }
                                display(d);
                                printf("Enter the values for x,y,z\n");
                                scanf("%d%d%d",&x,&y,&z);
```

```
                        res=eval();
                        printf("Result=%d\n",res);
                        break;
                case 2:printf("Enter the no. of terms in polynomial A\n");
                        scanf("%d",&n);
                        for(i=0;i<n;i++)
                        {
                                printf("Enter coefficient,exponent of x,y,z\n");
                                scanf("%d%d%d%d",&co,&expox,&expoy,&expoz);
                                a=attach(a,co,expox,expoy,expoz);
                        }
                        printf("Enter the no of terms in polynomial B\n");
                        scanf("%d",&m);
                        for(i=0;i<m;i++)
                        {
                                printf("Enter coeffients,exponents of x,y,z\n");
                                scanf("%d%d%d%d",&co,&expox,&expoy,&expoz);
                                b=attach(b,co,expox,expoy,expoz);
                        }
                        c=padd();
                        printf("Polynomial A:");
                        display(a);
                        printf("\nPolynomial B:");
                        display(b);
                        printf("\nResultant Polynomial:");
                        display(c);
                        printf("\n");
                        break;
                case 3:exit(0);
                default:printf("Invalid choice\n");
                }
        }
        return 0;
}
node *attach(node *p,int co,int expox,int expoy,int expoz)
{
        node *temp;
        temp=(node*)malloc(sizeof(node));
        temp->coef=co;
        temp->ex=expox;
        temp->ey=expoy;
        temp->ez=expoz;
        if(p==NULL)
                temp->link=temp;
        else
        {
                temp->link=p->link;
```

```
                p->link=temp;
        }
        p=temp;
        return p;
}
void display(node *p)
{
        node *temp;
        temp=p->link;
        while(temp!=p)
        {
                printf("%dx^%dy^%dz^%d+",temp->coef,temp->ex,temp->ey,temp->ez);
                temp=temp->link;
        }
        printf("%dx^%dy^%dz^%d\n",temp->coef,temp->ex,temp->ey,temp->ez);
}
int eval()
{
        node *temp;
        int sum=0;
        temp=d->link;
        while(temp!=d)
        {
                sum=sum+temp->coef*pow(x,temp->ex)*pow(y,temp->ey)*pow(z,temp->ez);
                temp=temp->link;
        }
        sum=sum+temp->coef*pow(x,temp->ex)*pow(y,temp->ey)*pow(z,temp->ez);
        return sum;
}
int compare(int x1,int x2,int y1,int y2,int z1,int z2)
{
        if((x1==x2)&&(y1==y2)&&(z1==z2))
                return 0;
        if(x1>x2)
                return 1;
        if(x1<x2)
                return -1;
        if(y1>y2)
                return 1;
        if(y1<y2)
                return -1;

        if(z1>z2)
                return 1;
        return -1;
}
node *padd()
```

```
{
        int i=0,j=0;
        node *starta,*startb;
        starta=a->link;
        startb=b->link;
        while((i<n)&&(j<m))
        {
                switch(compare(starta->ex,startb->ex,starta->ey,startb->ey,starta->ez,startb->ez))
                {
                        case 0:
                                c=attach(c,(starta->coef+startb->coef),starta->ex,starta->ey,starta-
>ez);
                                starta=starta->link;
                                startb=startb->link;
                                i++;
                                j++;
                                break;
                        case 1:
                                c=attach(c,starta->coef,starta->ex,starta->ey,starta->ez);
                                starta=starta->link;
                                i++;
                                break;
                        case -1:
                                c=attach(c,startb->coef,startb->ex,startb->ey,startb->ez);
                                startb=startb->link;
                                j++;
                                break;
                }
        }
        while(i<n)
        {
                c=attach(c,starta->coef,starta->ex,starta->ey,starta->ez);
                i++;
                starta=starta->link;
        }
        while(j<n)
        {
                c=attach(c,startb->coef,startb->ex,startb->ey,startb->ez);
                j++;
                startb=startb->link;
        }
        return c;
}
```

Output:

Polynomial operations
1.Evaluation
2.Addition of two polynomials

```
3.Exit
Enter your choice
1
Enter the no. of terms in polynomial
2
Enter coefficient,exponent of x,y,z
4 2 1 2
Enter coefficient,exponent of x,y,z
3 1 2 3
4x^2y^1z^2+3x^1y^2z^3
Enter the values for x,y,z
1 2 3
Result=396
Polynomial operations
1.Evaluation
2.Addition of two polynomials
3.Exit
Enter your choice
2
Enter the no. of terms in polynomial A
2
Enter coefficient,exponent of x,y,z
10 3 2 1
Enter coefficient,exponent of x,y,z
5 2 2 3
Enter the no of terms in polynomial B
2
Enter coeffients,exponents of x,y,z
15 3 2 1
Enter coeffients,exponents of x,y,z
7 1 2 3
Polynomial A:10x^3y^2z^1+5x^2y^2z^3

Polynomial B:15x^3y^2z^1+7x^1y^2z^3

Resultant Polynomial:25x^3y^2z^1+5x^2y^2z^3+7x^1y^2z^3

Polynomial operations
1.Evaluation
2.Addition of two polynomials
3.Exit
Enter your choice
3
```

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

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
        struct node *left;
        int data;
        struct node *right;
};
typedef struct node tree;
tree *root=NULL;
tree* create(tree *ptr, int data)
{
        if (ptr == NULL)
        {
                tree *temp;
                temp= (tree*)malloc(sizeof(tree));
                temp->data = data;
                temp->left = temp->right = NULL;
                return temp;
        }
        if (data < (ptr->data))
                ptr->left = create(ptr->left, data);

        else if (data > ptr->data)
                ptr -> right = create(ptr->right, data);
        return ptr;
}
tree* search(tree *ptr, int data)
{
        if(ptr == NULL)
                printf("\nElement not found");
        else if(data < ptr->data)
                ptr->left=search(ptr->left, data);
        else if(data > ptr->data)
                root->right=search(ptr->right, data);
        else
                printf("\nElement found is: %d", ptr->data);
        return ptr;
}
void inorder(tree *ptr)
```

```
{
        if(ptr)
        {
                inorder(ptr->left);
                printf("%d\t", ptr->data);
                inorder(ptr->right);
        }
}
void preorder(tree *ptr)
{
        if(ptr )
        {
                printf("%d\t", ptr->data);
                preorder(ptr->left);
                preorder(ptr->right);
        }
}
void postorder(tree *ptr)
{
        if(ptr)
        {
                postorder(ptr->left);
                postorder(ptr->right);
                printf("%d\t", ptr->data);
        }
}
void main()
{
        int data, ch, i, n;
        while (1)
        {
                printf("1. Create\n2.Inorder traversal\n3.Preorder traversal\n4.Postorder
        traversal\n5. Search\n6.Exit\n");
                printf("\nEnter your choice: ");
                scanf("%d", &ch);
                switch (ch)
                {
                        case 1: printf("Enter the number of elements \n " );
                                scanf("%d", &n);
                                printf("Enter the values to create BST \n");
                                for(i=0; i<n; i++)
                                {
                                        scanf("%d", &data);
                                        root=create(root, data);
                                }
                                break;
                        case 2: printf("Inorder Traversal: \n");
```

```
                        inorder(root);
                        break;
                case 3: printf("Preorder Traversal: \n");
                        preorder(root);
                        break;
                case 4: printf("Postorder Traversal: \n");
                        postorder(root);
                        break;
                case 5: printf("Enter the element to search\n ");
                        scanf("%d", &data);
                        root=search(root, data);
                        break;
                case 6: exit(0);
                default:printf("Wrong option\n");
                        break;
            }
        }
}
```

Output:

```
1. Create
2.Inorder traversal
3.Preorder traversal
4.Postorder traversal
5. Search
6.Exit

Enter your choice: 1
Enter the number of elements
 12
Enter the values to create BST
6 9 5 2 8 15 24 14 7 8 5 2
1. Create
2.Inorder traversal
3.Preorder traversal
4.Postorder traversal
5. Search
6.Exit

Enter your choice: 2
Inorder Traversal:
2       5       6       7       8       9       14      15      24
1. Create
2.Inorder traversal
3.Preorder traversal
4.Postorder traversal
5. Search
```

```
6.Exit

Enter your choice: 3
Preorder Traversal:
6       5       2       9       8       7       15      14      24
1. Create
2.Inorder traversal
3.Preorder traversal
4.Postorder traversal
5. Search
6.Exit

Enter your choice: 4
Postorder Traversal:
2       5       7       8       14      24      15      9       6
1. Create
2.Inorder traversal
3.Preorder traversal
4.Postorder traversal
5. Search
6.Exit

Enter your choice: 5
Enter the element to search
 14

Element found is: 14
1. Create
2.Inorder traversal
3.Preorder traversal
4.Postorder traversal
5. Search
6.Exit

Enter your choice: 5
Enter the element to search
 100

Element not found
1. Create
2.Inorder traversal
3.Preorder traversal
4.Postorder traversal
5. Search
6.Exit

Enter your choice: 6
```

11. Design, Develop and Implement a Program in C for the following operations on Graph(G) of Cities
a. Create a Graph of N cities using Adjacency Matrix.
b. Print all the nodes reachable from a given starting node in a digraph using BFS method
b. c. Check whether a given graph is connected or not using DFS method.

```
#include<stdio.h>

int a[10][10], n, i, j, source, visited[10],flag;
void create()
{
	printf("\nEnter the number of vertices of the digraph: ");
	scanf("%d", &n);
	printf("\nEnter the adjacency matrix of the graph:\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;
}
void dfs(int source)
{
	int v;
	visited[source] = 1;
	for(v=1; v<=n; v++)
		if(a[source][v] == 1 && visited[v] == 0)
		{
			printf("\n%d -> %d", source, v);
			dfs(v);
		}
}
int main()
{
	create();
	printf("\nEnter the source vertex to find the connectivity: ");
	scanf("%d",&source);
	flag=1;
	dfs(source);
	for(i=1;i<=n;i++)
		if(visited[i]==0)
			flag=0;

	if(flag==1)
		printf("\nGraph is Connected");
	else
		printf("\nGraph is not Connected");
	return 0;
```

```
}
```

Output:

```
Enter the number of vertices of the digraph: 5

Enter the adjacency matrix of the graph:
0 1 1 0 0
0 0 0 1 1
0 0 0 1 0
1 0 0 0 1
0 0 0 0 0

Enter the source vertex to find the connectivity: 1

1 -> 2
2 -> 4
4 -> 5
1 -> 3
Graph is Connected

[root@localhost Desktop]# ./a.out

Enter the number of vertices of the digraph: 5

Enter the adjacency matrix of the graph:
0 1 1 0 0
0 0 0 1 1
0 0 0 1 0
1 0 0 0 1
0 0 0 0 0

Enter the source vertex to find the connectivity: 5

Graph is not Connected
```

12. Given a File of N employee records with a set K of Keys(4-digit) which uniquely determine the records in file F. Assume that file F is maintained in memory by a Hash Table(HT) of m memory locations with L as the set of memory addresses (2-digit) of locations in HT. Let the keys in K and addresses in L are Integers. Design and develop a Program in C that uses Hash function H: K ®L as H(K)=K 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>

int key[20],n,m;
int *ht=NULL;
int ind;
int count = 0;

void insert(int key)
{
   ind = key % m;
   while(ht[ind] != -1)
       ind = (ind+1)%m;
   ht[ind] = key;
   count++;
 }
void display()
{
   int i;
   if(count == 0)
   {
      printf("Hash Table is empty\n");
      return;
   }

   printf("Hash Table contents are:\n ");
   printf("T[i]\tHash Key\n");
   for(i=0; i<m; i++)
      printf("T[%d]\t%d\n ", i, ht[i]);
}
void main()
{
   int i;
   printf("Enter the number of employee  records (N) \n");
   scanf("%d", &n);

   printf("Enter the two digit memory locations (m) for hash table\n   ");
   scanf("%d", &m);
```

```
    ht = (int *)malloc(m*sizeof(int));
    for(i=0; i<m; i++)
      ht[i] = -1;
printf("Enter the four digit key values (K) for N Employee Records:\n  ");
          for(i=0; i<n; i++)
       scanf("%d", &key[i]);

    for(i=0;i<n;i++)
    {
       if(count == m)
       {
          printf("Hash table is full. Cannot insert the record %d key\n",i+1);
          break;
       }
       insert(key[i]);
    }
    display();
}
```

Output:

```
Enter the number of employee  records (N)
5
Enter the two digit memory locations (m) for hash table
10
Enter the four digit key values (K) for N Employee Records:
1288
1388
1499
1500
7545
Hash Table contents are:
T[i]     Hash Key
T[0]    1499
 T[1]   1500
 T[2]   -1
 T[3]   -1
 T[4]   -1
 T[5]   7545
 T[6]   -1
 T[7]   -1
 T[8]   1288
 T[9]   1388
```