DATA STRUCTURES LAB MANUAL IN C PROGRAMMING LANGUAGE

YouTube Channel Link: Vishwanath M

Data Structures Playlist:

Vishwanath M - Data Structures

LAB 1

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

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

Program:

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

#define NO_OF_DAYS 7

typedef struct
{
        char* name_of_day;
        int date_of_day;
        char* activity_descr;
} CALENDAR;

void create_calendar(CALENDAR a[], int i, char name[], int date, char activity[])
{
        a[i].name_of_day = (char *) malloc(strlen(name) + 1);
        strcpy(a[i].name_of_day, name);

        a[i].date_of_day = date;

        a[i].activity_descr = (char *) malloc(strlen(activity) + 1);
        strcpy(a[i].activity_descr, activity);
}

void read_calendar(CALENDAR a[])
{
        int i, date;
        char name[10], activity[10];
        for (i = 0; i < NO_OF_DAYS; i++)
        {
```

```
                scanf("%s", name);
                scanf("%d", &date);
                scanf("%s", activity);
                create_calendar(a, i, name, date, activity);
        }
}

void print_weeks_activity(CALENDAR a[])
{
        int i;
        printf("\nWeeks activity\n");
        for (i = 0; i < NO_OF_DAYS; i++)
        {
                printf("\%s: %s\n", a[i].name_of_day, a[i].activity_descr);
        }
}

void main()
{
        CALENDAR a[NO_OF_DAYS];
        printf("Enter Day Date Activity\n\n");
        read_calendar(a);
        print_weeks_activity(a);
}
```

LAB 2

2. Develop a Program in C for the following operations on Strings.
    a. Read a main String (STR), a Pattern String (PAT) and a Replace String (REP)
    b. Perform Pattern Matching Operation: Find and Replace all occurrences of PAT in STR with REP if PAT exists in STR. Report suitable messages in case PAT does not exist in STR.

    Support the program with functions for each of the above operations. Don't use Built-in functions.

Program:

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

                        }
                }
                else
                {
                        ans[j]= str[c];
                        j++;
                        c++;
                        m=c;
                        i=0;
                }
```

```
        }
        ans[j]='\0';
}

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

LAB 3

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

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

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

int a[MAX];
int top=-1;

void push(int item)
{
        if(top==(MAX-1))
        {
                printf("STACK OVERFLOW\n");
                return;
        }
        top=top+1;
        a[top]=item;
}

int pop()
{
        if(top==-1)
        {
                printf("STACK UNDERFLOW\n");
                return -1;
        }
        int item=a[top];
        top=top-1;
        return item;
}

void display()
{
        int i;
```

```
	if(top==-1)
	{
		printf("STACK IS EMPTY");
		return;
	}
	printf("STACK ELEMENTS ARE:\n");
	for(i=top;i>=0;i--)
	{
		printf("%d ",a[i]);
	}
	printf("\n");
}

void cpalindrome()
{
	int flag=1;
	int i;

	printf("STACK ELEMENTS ARE:\n");
	for( i=top;i>=0;i--)
	{
		printf("%d ",a[i]);
	}
	printf("\nREVERSE STACK ELEMENTS ARE:\n");
	for( i=0;i<=top;i++)
	{
		printf("%d ",a[i]);
	}

	for(i=0; i<=top/2; i++)
	{
		if(a[i]!=a[top-i])
		{
			flag=0;
			break;
		}
	}

	if(flag==1)
	{
		printf("\n It is a Palindrome\n");

	}
```

```
        else
        {
                printf("\n It is not a Palindrome\n");
        }


}

void main()
{
        int choice,item;

        for(;;)
        {
                printf("\nMENU\n");
                printf("press 1:Push\t2:pop\t3:display\t4:cpalindrome\t5:exit\n");
                printf("Enter your choice: ");
                scanf("%d",&choice);
                switch(choice)
                {

                        case 1: printf("enter the item to be inserted\n");
                                scanf("%d",&item);
                                push(item);
                                break;

                        case 2: item=pop();
                                if(item!=-1)
                                        printf("item popped: %d\n",item);
                                break;

                        case 3: display();
                                break;

                        case 4: cpalindrome();
                                break;

                        case 5: exit(0);

                }
        }

}
```

LAB 4

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

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

char stack[100];
int top=-1;

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

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

int priority(char x)
{
        if(x=='(')                      return 1;
        if(x=='+'||x=='-')              return 2;
        if(x=='*'||x=='/'||x=='%')      return 3;
        if(x=='^'||x=='$')              return 4;
        return 1;
}

void main()
{
        char expression[100];
        char symbol,a;
        int i=0;

        printf("enter infix expression:\n");
        scanf("%s",expression);

        printf("postfix expression is\n");
```

```
        for(i=0;i<strlen(expression);i++)
        {
                symbol=expression[i];
                if(isalnum(symbol))
                        printf("%c ",symbol);
                else if(symbol=='(')
                        push(symbol);
                else if(symbol==')')
                {
                        while((a=pop())!='(') printf("%c ",a);
                }
                else
                {
                        while(priority(stack[top])>=priority(symbol))
                        {
                                printf("%c ",pop());
                        }
                        push(symbol);
                }
        }
        while(top!=-1)
        {
                printf("%c ",pop());
        }
        printf("\n");
}
```

LAB 5

5. Develop 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

5A

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

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

void main()
{
	char expression[20];
	int op1,op2,i,top=-1,result,final;
	char symbol;
	int stack[20];
	printf("enter postfix expression:\n");
	scanf("%s",expression);
	for(i=0;i<strlen(expression);i++)
	{
		symbol=expression[i];
		if(isdigit(symbol))
			stack[++top]=symbol-'0';
		else
		{
			op2=stack[top--];
			op1=stack[top--];
```

```
                        result=compute(symbol,op1,op2);
                        stack[++top]=result;
                }
        }
        final=stack[top--];
        printf("value of expression: %d\n",final);
}
```

5B

```
#include  <stdio.h>
#include <stdlib.h>
#include <math.h>
void TOH(int n,char source,char dest,char aux)
{
        if(n==1)
        {
                printf("Move disc %d from %c to %c\n", n,source,dest);
                return;
        }
        TOH(n-1,source,aux,dest);
        printf("Move disc %d from %c to %c\n", n,source,dest);
        TOH(n-1,aux,dest,source);
}
void main()
{
        int n;
        printf("Enter no of discs\n");
        scanf("%d",&n);
        int count = pow(2,n)-1;
        printf("Number of Moves: %d\n",count);
        TOH(n,'A','B','C');
}
```

LAB 6

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

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

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

int count=0;
char cq[cqsize];
int f=0,r=-1;

void insert(char item)
{
        if(count==cqsize)
        {
                printf("CQ OVERFLOW!\n");
                return;
        }
        r=(r+1)%cqsize;
        cq[r]=item;
        count++;
        printf("item inserted is %c\n",item);

}
void delete()
{
        if(count==0)
        {
                printf("CQ UNDERFLOW!\n");
                return;
        }
        char itemdel=cq[f];
        count--;
        printf("item deleted is %c\n",itemdel);

        if(f==r) // ONLY ONE ELEMET
        {
```

```
                f=0;    //queue reset
                r=-1;
        }
        f=(f+1)%cqsize;
}
void display()
{
        if(count==0)
        {
                printf("CQ EMPTY!\n");
                return;
        }

        printf("f: %d, r: %d\n",f,r);

        int t=f;     // queue printed from first always
        printf("CQ CONTENTS:\n");
        for(int i=1;i<=count;i++)
        {
                printf("%c ",cq[t]);
                t=(t+1)%cqsize;
        }
        printf("\n");

}

void main()
{
        char item;
        int choice;
        printf("1:insert 2:delete 3:display 4:exit\n");
        for(;;)
        {
                printf("\nenter your choice: ");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1: printf("enter item to insert: ");
                        scanf(" %c",&item);
                        insert(item);
                        break;

                        case 2: delete();
                        break;
```

```
                    case 3: display();
                    break;

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

LAB 7

7. Develop a menu driven Program in C for the following operations on Singly Linked List (SLL) of Student Data with the fields: USN, Name, Programme, Sem, PhNo.
    a. Create a SLL of N Students Data by using front insertion.
    b. Display the status of SLL and count the number of nodes in it
    c. Perform Insertion / Deletion at End of SLL
    d. Perform Insertion / Deletion at Front of SLL(Demonstration of stack)
    e. Exit

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

struct node
{
        char usn[25],name[25],branch[25];
        int sem;
        long int phone;
        struct node* link;
};
typedef struct node* NODE;

NODE start = NULL;
int count=0;

NODE create()
{
        NODE snode;
        snode = (NODE)malloc(sizeof(struct node));
        if(snode == NULL)
        {
                printf("\nMemory is not available");
                exit(1);
        }
        printf("\nEnter the usn,Name,Branch, sem,PhoneNo of the student:");
        scanf("%s %s %s %d %ld",snode->usn, snode->name,
snode->branch,&snode->sem,&snode->phone);
        snode->link=NULL;
        count++;
        return snode;
}

NODE insertfront()
{
        NODE temp;
```

```
        temp = create();
        if(start == NULL)
        {
                return temp;
        }
        temp->link = start;   //new node will become the start node
        return temp;
}

NODE deletefront()
{
        NODE temp;
        if(start == NULL)
        {
                printf("\nLinked list is empty");
                return NULL;
        }
        if(start->link == NULL) // only start node is present in the SLL
        {
                printf("\nThe Student node with usn:%s is deleted ",start->usn);
                count--;
                free(start);   // free is memory de-allocation function
                return NULL;
        }

        temp = start;
        start = start->link;  // start->link is made as next 'start'
        printf("\nThe Student node with usn:%s is deleted",temp->usn);
        count--;
        free(temp);
        return start;
}

NODE insertend()
{
        NODE cur,temp;
        temp = create();
        if(start == NULL)  // empty SLL
        {
                return temp;
        }
        cur = start;
        while(cur->link !=NULL) // Till last node
        {
```

```
                cur = cur->link;
        }
        cur->link = temp;
        return start;
}

NODE deleteend()
{
        NODE cur,prev;
        if(start == NULL)
        {
                printf("\nLinked List is empty");
                return NULL;
        }
        if(start->link == NULL) // single node
        {
                printf("\nThe student node with the usn:%s is deleted",start->usn);
                free(start);
                count--;
                return NULL;
        }
        prev = NULL;
        cur = start;
        while(cur->link!=NULL) // Till last node
        {
                prev = cur;
                cur = cur->link;
        }
        printf("\nThe student node with the usn:%s is deleted",cur->usn);
        free(cur);
        prev->link = NULL;  // last but one node will not be connected to anything
        count--;
        return start;
}

void display()
{
        NODE cur;
        int num=1;
        if(start == NULL)
        {
                printf("\nNo Contents to display in SLL \n");
                return;
        }
```

```
        printf("\nThe contents of SLL: \n");
        cur = start;
        while(cur!=NULL)
        {
        printf("\n||%d|| USN:%s| Name:%s| Branch:%s| Sem:%d| Ph:%ld|",num,cur->usn,
cur->name,cur->branch, cur->sem,cur->phone);
        cur = cur->link;
        num++;
        }
        printf("\n No of student nodes is %d \n",count); // count is global variable
}

void stackdemo()  // choice 5 of main function
{
        int ch;
        while(1)
        {
                printf("\n~~~Stack Demo using SLL~~~\n");
                printf("\n1:Push operation \n2: Pop operation \n3: Display \n");
                printf("\nEnter your choice for stack demo");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1: start = insertfront();
                                break;
                        case 2: start = deletefront();
                                break;
                        case 3: display();
                                break;
                        default : return;
                }
        }
        return;
}


int main()
{
        int ch,i,n;
        while(1)
        {
                printf("\n~~~Menu~~~");
                printf("\nEnter your choice for SLL operation \n");
                printf("\n1:Create SLL of Student Nodes");
                printf("\n2:Display Status");
```

```
                printf("\n3:Insert At End");
                printf("\n4:Delete At End");
                printf("\n5:Stack Demo using SLL(Insertion and Deletion at Front)");
                printf("\n6:Exit \n");

                printf("\nEnter your choice:");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1 :
                                printf("\nEnter the no of students:");
                                scanf("%d",&n);
                                for(i=1;i<=n;i++)
                                        start = insertfront();
                                break;
                        case 2:
                                display();
                                break;
                        case 3:
                                start = insertend();
                                break;

                        case 4:
                                start = deleteend();
                                break;
                        case 5:
                                stackdemo();
                                break;
                        case 6:
                                exit(0);

                        default: printf("\n Please enter the valid choice");
                }
        }
}
```

LAB 8

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

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

struct node
{
        char ssn[25],name[25],dept[25],desgn[25];
        int sal;
        long int phone;
        struct node* llink;
        struct node* rlink;
};
typedef struct node* NODE;

NODE start = NULL;
int count=0;

NODE create()
{
        NODE snode;
        snode = (NODE)malloc(sizeof(struct node));
        if(snode == NULL)
        {
                printf("\nMemory is not available");
                exit(1);
        }
        printf("\nEnter the ssn,Name,dept,designation, sal,PhoneNo of the employee:");
        scanf("%s %s %s %s %d %ld",snode->ssn, snode->name, snode->dept,snode->desgn,
&snode->sal,&snode->phone);
        snode->llink=NULL;
        snode->rlink=NULL;
        count++;
        return snode;
}
```

```
NODE insertfront()
{
        NODE temp;
        temp = create();
        if(start == NULL)
        {
                return temp;
        }
        temp->rlink = start;
        start->llink= temp;
        return temp;
}

NODE deletefront()
{
        NODE temp;
        if(start == NULL)
        {
                printf("\nLinked list is empty");
                return NULL;
        }
        if(start->rlink == NULL) // only start node is present in the DLL
        {
                printf("\nThe Employee node with ssn:%s is deleted ",start->ssn);
                count--;
                free(start);   // free is memory de-allocation function
                return NULL;
        }

        temp = start;
        start = start->rlink;  // start->link is made as next 'start'
        temp->rlink = NULL;
        start->llink = NULL;
        printf("\nThe Employee node with ssn:%s is deleted",temp->ssn);
        count--;
        free(temp);
        return start;
}

NODE insertend()
{
        NODE cur,temp;
        temp = create();
        if(start == NULL)  // empty DLL
```

```
        {
                return temp;
        }
        cur = start;
        while(cur->rlink !=NULL) // Till last node
        {
                cur = cur->rlink;
        }
        cur->rlink = temp;
        temp->llink=cur;
        return start;
}

NODE deleteend()
{
        NODE cur,prev;
        if(start == NULL)
        {
                printf("\nLinked List is empty");
                return NULL;
        }
        if(start->rlink == NULL) // single node
        {
                printf("\nThe Employee node with the ssn:%s is deleted",start->ssn);
                free(start);
                count--;
                return NULL;
        }
        prev = NULL;
        cur = start;
        while(cur->rlink!=NULL) // Till last node
        {
                prev = cur;
                cur = cur->rlink;
        }
        printf("\nThe employee node with the ssn:%s is deleted",cur->ssn);
        prev->rlink = NULL;   // last but one node will not be connected to anything
        cur->llink=NULL;
        free(cur);
        count--;
        return start;
}

void display()
```

```
{
        NODE cur;
        int num=1;
        if(start == NULL)
        {
                printf("\nNo Contents to display in DLL \n");
                return;
        }
        printf("\nThe contents of DLL: \n");
        cur = start;
        while(cur!=NULL)
        {
        printf("\n||%d|| SSN:%s| Name:%s| Dept:%s| Desgn:%s| Sal:%d| Ph
No:%ld|",num,cur->ssn, cur->name,cur->dept, cur->desgn, cur->sal,cur->phone);
        cur = cur->rlink;
        num++;
        }
        printf("\n No of Employee nodes is %d \n",count); // count is global variable
}

void dequeuedemo()  // choice 7 of main function
{
        int ch;
        while(1)
        {
                printf("\n~~~DE-Queue Demo using DLL~~~\n");
                printf("\n1:Enqueue front \n2: Dequeue front \n3: Enqueue end \n4: Dequeue
end\n5: Display \n");
                printf("\nEnter your choice for DE-queue demo");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1: start = insertfront();
                                break;
                        case 2: start = deletefront();
                                break;
                        case 3: start= insertend();
                                break;
                        case 4: start = deleteend();
                                break;
                        case 5: display();
                                break;
                        default : return;
                }
```

```
        }
        return;
}

int main()
{
        int ch,i,n;
        while(1)
        {
                printf("\n~~~Menu~~~");
                printf("\nEnter your choice for DLL operation \n");
                printf("\n1:Create DLL of Employee Nodes");
                printf("\n2:Display Status");
                printf("\n3:Insert At End");
                printf("\n4:Delete At End");
                printf("\n5:Insert At front");
                printf("\n6:Delete At Front");
                printf("\n7:DE-Queue Demo using DLL(Insertion and Deletion at Both Ends)");
                printf("\n8:Exit \n");

                printf("\nEnter your choice:");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1 :
                                printf("\nEnter the no of Employees:");
                                scanf("%d",&n);
                                for(i=1;i<=n;i++)
                                        start = insertfront();
                                break;
                        case 2:
                                display();
                                break;
                        case 3:
                                start = insertend();
                                break;

                        case 4:
                                start = deleteend();
                                break;
                        case 5:
                                start= insertfront();
                                break;
                        case 6:
```

```
                        start= deletefront();
                        break;
                case 7:
                        dequeuedemo();
                        break;
                case 8:
                        exit(0);

                default: printf("\n Please enter the valid choice");
        }
    }
}
```

LAB 9

9. Develop a Program in C for the following operationson Singly Circular Linked List (SCLL) with header nodes

a. Represent and Evaluate a Polynomial P(x,y,z) = 6x 2 y 2 z-4yz 5 +3x 3 yz+2xy 5 z-2xyz 3

b. Find the sum of two polynomials POLY1(x,y,z) and POLY2(x,y,z) and store the result in POLYSUM(x,y,z)

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

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
struct node
{
        int coef,flag;
        int xexp,yexp,zexp;
        struct node *link;
};
typedef struct node* NODE;

NODE create()
{
        NODE p;
        p=(NODE)malloc(sizeof(struct node));
        if(p==NULL)
        {
                printf("INSUFFIECIENT MEMORY\n");
                exit(0);
        }
        return p;
}

NODE attach(int coef,int xexp,int yexp,int zexp,NODE head)
{
        NODE temp,cur;
        temp=create();
        temp->coef=coef;
        temp->xexp=xexp;
        temp->yexp=yexp;
        temp->zexp=zexp;
        temp->flag=0;
        cur=head->link;

        while(cur->link!=head)
```

```
        {
                cur=cur->link;
        }
        cur->link=temp;
        temp->link=head;
        return head;
}

NODE read_poly(NODE head)
{
        int xexp,yexp,zexp,coef;
        int i,n;
        printf("enter no of terms in the polynomial: ");
        scanf("%d",&n);
        for(i=1;i<=n;i++)
        {
                printf("\n\tenter %dth term",i);
                printf("\n\tenter coef,xexp,yexp,zexp\n\t");
                scanf("%d%d%d%d",&coef,&xexp,&yexp,&zexp);
                head=attach(coef,xexp,yexp,zexp,head);
        }
        return head;
}
void display(NODE head)
{
        NODE temp;
        if(head->link==head)
        {
                printf("polynomial doesnt exist\n");
                return;
        }
        temp=head->link;
        while(temp!=head)
        {
                printf("%dx^%dy^%dz^%d",temp->coef,temp->xexp,temp->yexp,temp->zexp);
                temp=temp->link;
                if(temp!=head) printf(" + ");
        }
}
void evaluate(NODE head)
{
        int x,y,z,val=0;
        NODE poly;
        printf("\nenter value of x,y,z: ");
```

```
        scanf("%d%d%d",&x,&y,&z);
        poly=head->link;
        while(poly!=head)
        {
                val=val+((poly->coef)*pow(x,poly->xexp)*pow(y,poly->yexp)*pow(z,poly->zexp));
                poly=poly->link;
        }
        printf("evaluation result: %d\n",val);
}

NODE polyadd(NODE h1,NODE h2,NODE h3)
{
        NODE p1,p2;
        int x1,y1,z1,coef1;
        int x2,y2,z2,coef2;
        int coefres;
        p1=h1->link;
        while(p1!=h1)
        {
                int considered=0;
                x1=p1->xexp;
                y1=p1->yexp;
                z1=p1->zexp;
                coef1=p1->coef;

                p2=h2->link;
                while(p2!=h2)
                {
                        x2=p2->xexp;
                        y2=p2->yexp;
                        z2=p2->zexp;
                        coef2=p2->coef;

                        if(x1==x2&&y1==y2&&z1==z2)
                        {
                                coefres=coef1+coef2;
                                considered=1;
                                p2->flag=1;
                                if(coefres!=0)  // if coeffs add to 0, need not be displayed
                                        h3=attach(coefres,x1,y1,z1,h3);
                        }
                        p2=p2->link;
                }
                if(considered==0)
```

```
                        h3=attach(coef1,x1,y1,z1,h3);
                p1=p1->link;
        }

        p2=h2->link;
        while(p2!=h2)
        {
                if(p2->flag==0)  // not already considered for addition
                {
                        h3=attach(p2->coef,p2->xexp,p2->yexp,p2->zexp,h3);  //add tailing terms
                }
                p2=p2->link;
        }
        return h3;
}

void main()
{
        NODE head,h1,h2,h3;
        int choice;
        head=create();
        h1=create();
        h2=create();
        h3=create();

        // making them Singly CIRCULAR Linked list nodes
        head->link=head;
        h1->link=h1;
        h2->link=h2;
        h3->link=h3;

        while(1)
        {
                printf("\n1: evaluation 2: addition 3: exit\n");
                printf("enter your choice: ");
                scanf("%d",&choice);
                switch(choice)
                {
                case 1:
                        printf("polynomial evaluation\n");
                        head=read_poly(head);
                        display(head);
                        evaluate(head);
                        break;
```

```
                case 2:
                        printf("\nenter poly1:\n");
                        h1=read_poly(h1);
                        display(h1);

                        printf("\nenter poly2:\n");
                        h2=read_poly(h2);
                        display(h2);

                        h3=polyadd(h1,h2,h3);
                        printf("\npoly addition result:\n");
                        display(h3);
                        break;

                case 3: exit(0);
                }
        }
}
```

LAB 10

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

```
#include<stdio.h>
#include<stdlib.h>
struct BST
{
        int data;
        struct BST* lchild;
        struct BST*rchild;
};

typedef struct BST* NODE;

NODE create()
{
        NODE temp;
        temp = (NODE) malloc(sizeof(struct BST));
        printf("Enter The value: ");
        scanf("%d", &temp->data);
        temp->lchild = NULL;
        temp->rchild = NULL;
        return temp;
}

void insert(NODE root, NODE newnode)
{
        if (newnode->data < root->data)
        {
                if (root->lchild == NULL)
                        root->lchild = newnode;
                else
                        insert(root->lchild, newnode);
        }


        if (newnode->data > root->data)
        {
                if (root->rchild == NULL)
                        root->rchild = newnode;
```

```
                else
                        insert(root->rchild, newnode);
        }
}


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

void inorder(NODE root)
{
        if(root != NULL)
        {
                inorder(root->lchild);
                printf("%d ", root->data);
                inorder(root->rchild);
        }
}

void preorder(NODE root)
{
```

```
        if (root != NULL)
        {
                printf("%d ", root->data);
                preorder(root->lchild);
                preorder(root->rchild);
        }
}

void postorder(NODE root)
{
        if (root != NULL)
        {
                postorder(root->lchild);
                postorder(root->rchild);
                printf("%d ", root->data);
        }
}

void main()
{
        int ch, i, n;
        NODE root = NULL, newnode;
        while(1)
        {
                printf("\n\n~~~~BST MENU~~~~");
                printf("\n1.Create a BST");
                printf("\n2.Search");
                printf("\n3.BST Traversals: ");
                printf("\n4.Exit");
                printf("\nEnter your choice: ");
                scanf("%d", &ch);
                switch(ch)
                {
                        case 1:
                        printf("\nEnter the number of elements: ");
                        scanf("%d", &n);
                        for(i=1;i<=n;i++)
                        {
                                newnode = create();
                                if (root == NULL)
                                        root = newnode;
                                else
                                        insert(root, newnode);
                        }
```

```
                        break;

                        case 2:
                        search(root);
                        break;

                        case 3:
                        if (root == NULL)
                                printf("\nTree Is Not Created");
                        else
                        {
                                printf("\nThe Preorder display : ");
                                preorder(root);
                                printf("\nThe Inorder display : ");
                                inorder(root);
                                printf("\nThe Postorder display : ");
                                postorder(root);
                        }
                        break;


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

LAB 11

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

```
#include<stdio.h>
#include<stdlib.h>
int a[50][50], n, bvisited[50],dvisited[50];
int q[20], front = -1,rear = -1;
int s[20], top = -1, count=0;

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

void dfs(int v)
{
        int i;
        dvisited[v]=1;
        s[++top] = v;
        for(i=1;i<=n;i++)
        {
                if(a[v][i] == 1&& dvisited[i] == 0 )
                {
                        printf("%d ", i);
                        dfs(i);
                }
```

```
        }
}

int main()
{
        int ch, start, i,j;
        printf("\nEnter the number of vertices in graph: ");
        scanf("%d",&n);
        printf("\nEnter the adjacency matrix:\n");
        for(i=1; i<=n; i++)
        {
                for(j=1;j<=n;j++)
                scanf("%d",&a[i][j]);
        }
        for(i=1;i<=n;i++)
        {
                bvisited[i]=0;
                dvisited[i]=0;
        }
        printf("\nEnter the starting vertex: ");
        scanf("%d",&start);

        while(1)
        {
                printf("\nMENU");
                printf("\n==>1. BFS: Print all nodes reachable from a given starting node");
                printf("\n==>2. DFS: Print all nodes reachable from a given starting node");
                printf("\n==>3:Exit");

                printf("\nEnter your choice: ");
                scanf("%d", &ch);
                switch(ch)
                {
                        case 1: printf("\nNodes reachable from starting vertex %d are: ", start);
                                bfs(start);
                                break;

                        case 2: printf("\nNodes reachable from starting vertex %d are:",start);
                                dfs(start);
                                break;

                        case 3: exit(0);

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

```
                }
        }
}
```

LAB 12

12. Given a File of N employee records with a set K of Keys (4-digit) which uniquely determine the records in file F. Assume that file F is maintained in memory by a Hash Table (HT) of m memory locations with L as the set of memory addresses (2-digit) of locations in HT. Let the keys in K and addresses in L are Integers. Develop a Program in C that uses 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 empkey[20],hashindex,n,m,*ht,elecount=0;

void createhashtable()
{
	ht=(int*)malloc(m*sizeof(int));
	if(ht==NULL)
		printf("MEMORY UNAVAILABLE!");
	else
	{
		for(int i=0;i<m;i++)
		{
			ht[i]=-1;
		}
	}
}

void insertintohashtable(int key)
{
	hashindex=key%m;
	if(ht[hashindex]!=-1)
	{
		hashindex=(hashindex+1)%m;
	}
	ht[hashindex]=key;
	elecount++;
}

void display()
{
	int i;
	if(elecount==0)
	{
		printf("HASH TABLE EMPTY\n");
```

```
        }
        for(i=0;i<m;i++)
        {
                printf("T[%d]-->%d\n",i,ht[i]);
        }
}

void main()
{
        int i;
        printf("enter no of records: ");
        scanf("%d",&n);
        printf("enter hashtable size: ");
        scanf("%d",&m);
        printf("enter emp key values: \n");
        for(i=0;i<n;i++)
        {
                scanf("%d",&empkey[i]);
        }
        createhashtable();
        printf("inserting keys into hashtable:\n");
        for(i=0;i<n;i++)
        {
                if(elecount==m)
                {
                        printf("HASH TABLE FULL\n");
                        printf("cant insert %d key\n",empkey[i]);
                        break;
                }
                insertintohashtable(empkey[i]);
        }
        display();
}
```