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.

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

struct day{
    char *dayname;
    int d,m,y;
    char *activitydescription;
};

void create(struct day *calendar){
        char *daynames[]= \
                {"Monday","Tuesday","Wednesday","Thursday","Friday","Saturday","Sunday"};
        for(int i=0;i<7;i++){
                calendar[i].dayname=strdup(daynames[i]);
                size_t bufferSize=256;
                calendar[i].activitydescription=(char *)malloc(bufferSize*sizeof(char));
        }
}

void read(struct day *calendar){
        for(int i=0;i<7;i++){
                printf("Enter date for %s in dd/mm/yy : ",calendar[i].dayname);
                scanf("%d%d%d",&calendar[i].d,&calendar[i].m,&calendar[i].y);
                printf("Enter activity for %s : ",calendar[i].dayname);
                while(getchar()!='\n')
                    ;
                size_t bufferSize=256;
                getline(&calendar[i].activitydescription,&bufferSize,stdin);
        }
}

void display(struct day *calendar){
        printf("%-10s %-10s %-10s\n","Day","Date","Activity");
        for(int i=0;i<7;++i){
                printf("%-10s %d/%d/%d %-10s\n", \
                        calendar[i].dayname,calendar[i].d, \
                        calendar[i].m,calendar[i].y, \
                        calendar[i].activitydescription
                );
        }
}

int main(){
        struct day *calendar=(struct day *)malloc(7*sizeof(struct day));
        if(calendar==NULL){
                fprintf(stderr,"Memory allocation failed\n");
                return 1;
        }

        create(calendar);
        read(calendar);
        display(calendar);

        for(int i=0;i<7;++i){
                free(calendar[i].dayname);
                free(calendar[i].activitydescription);
        }
        free(calendar);
        return 0;
}
```

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.

→
```c
#include<stdio.h>

void main(){
        char STR[100],PAT[100],REP[100],ANS[100];
        int c,i,j,m,k,flag;
        printf("Enter the MAIN string : ");
        scanf("%[^\n]",STR);
        printf("Enter the PATTERN string : ");
        scanf("%s",PAT);
        printf("Enter the REPLACE string : ");
        scanf("%s",REP);
        c=i=j=m=k=flag=0;

        while(STR[c]!='\0'){
                if(STR[m]==PAT[i]){
                        i++;
                        m++;
                        flag=1;
                        if(PAT[i]=='\0'){
                                for(k=0;REP[k]!='\0';j++,k++)
                                        ANS[j]=REP[k];
                                i=0;
                                c=m;
                        }
                }else{
                        ANS[j]=STR[c];
                        j++;
                        c++;
                        m=c;
                        i=0;
                }
        }

        if(flag){
                ANS[j]='\0';
                printf("\nThe resultant string is %s\n",ANS);
        }else
                printf("Pattern doesn't found !\n");
}
```

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.

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

#define max_size 5

int stack[max_size],top=-1,i,item;

void push(){
        if(top==(max_size-1))
                printf("\nStack Overflow !");
        else{
                printf("Enter the element to be inserted : ");
                scanf("%d",&item);
                stack[++top]=item;
        }
}

void pop(){
        if(top==-1)
                printf("Stack Underflow !\n");
        else
                printf("\nThe poped element : %d\n",stack[top--]);
}

void pali(){
        if(top==-1)
                printf("Push some elements into the stack first !\n");
        else
                for(i=top;i>=0;i--)
                        if(stack[i]!=stack[top-i]){
                                printf("Not Palindrome\n");
                                return;
                        }

        printf("Palindrome !\n");
}

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

int main(){
        int choice;
        printf("\n\n--------STACK OPERATIONS--------\n");
        printf("1.Push\n2.Pop\n3.Palindrome\n4.Display\n5.Exit\n");
        while(1){
                printf("> ");
                scanf("%d",&choice);
                switch(choice){
                        case 1:
                                push();
                                break;
                        case 2:
                                pop();
                                break;
                        case 3:
                                pali();
                                break;
                        case 4:
                                display();
                                break;
                        case 5:
                                exit(0);
                                break;

                        default:
                                printf("\nInvalid choice !");
                                break;
                }
        }
}
```

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 :
   + - * / % ^ and alphanumeric operands

→
```c
#include <ctype.h>
#include <stdio.h>

#define SIZE 50

char s[SIZE],elem;
int top=-1;

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

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

int pr(char elem){
        switch(elem){
                case '#':
                        return 0;
                case '(':
                        return 1;
                case '+':
                case '-':
                        return 2;
                case '*':
                case '/':
                case '%':
                        return 3;
                case '^':
                        return 4;
        }
        return 0;
}

void main(){
        char infx[SIZE],pofx[SIZE],ch;
        int i=0,k=0;

        printf("\nRead the Infix Expression\n> ");
        scanf("%s",infx);
        push('#');
        while((ch=infx[i++])!='\0'){
                if(ch=='(')
                        push(ch);
                else if(isalnum(ch))
                        pofx[k++]=ch;
                else if(ch==')'){
                        while(s[top]!='(')
                                pofx[k++]=pop();
                        elem=pop();
                }else{
                        while(pr(s[top])>=pr(ch))
                                pofx[k++]=pop();
                        push(ch);
                }
        }
        while(s[top]!='#')
                pofx[k++]=pop();
        printf("\nGiven Infix expression : %s",infx);
        printf("\nPostfix expression     : %s\n",pofx);
}
```

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.

→ a.
```c
#include<stdio.h>
#include<ctype.h>
#include<string.h>

float compute(char symbol,float op1,float op2){
        switch(symbol){
                case '+' :
                        return op1+op2;
                case '-' :
                        return op1-op2;
                case '*' :
                        return op1*op2;
                case '/' :
                        return op1/op2;
                case '$' :
                        return op1+op2;
                default :
                        return 0;
        }
}

void main(){
        float s[20],res,op1,op2;
        int top,i;
        char postfix[20],symbol;

        printf("Enter the postfix expression:\n> ");
        scanf("%s",postfix);
        top=-1;

        for(i=0;i<strlen(postfix);i++){
                symbol=postfix[i];
                if(isdigit(symbol))
                        s[++top]=symbol-'0';
                else{
                        op2=s[top--];
                        op1=s[top--];
                        res=compute(symbol,op1,op2);
                        s[++top]=res;
                }
        }

        printf("The result is %f\n",s[top--]);
}
```

   b.
```c
#include<stdio.h>
#include<math.h>

void tower(int n,char beg,char aux,char end){
        if(n==0)
                printf("Illegal, Try with non-zero Positive Integers !\n");
        else if(n==1)
                printf("Move Disc from %c to %c\n",beg,end);
        else{
                tower(n-1,beg,end,aux);
                tower(1,beg,aux,end);
                tower(n-1,aux,beg,end);
        }
}

void main(){
        int n;
        printf("Enter the number of Discs :\n> ");
        scanf("%d",&n);
        printf("Tower of Hanoi for %d Disc has the following steps :\n",n);
        tower(n,'a','b','c');
        printf("\n\nTotal Number of moves are : %d\n",(int)pow(2,n)-1);
}
```

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

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

#define max 10

int q[10],front=0,rear=-1;

void insert(){
        int x;
        if((front==0&&rear==max-1)||(front>0&&rear==front-1))
                printf("Queue is overflow !\n");
        else{
                printf("Enter element to be insert : ");
                scanf("%d",&x);
                if(rear==max-1&&front>0){
                        rear=0;
                        q[rear]=x;
                }else if((front==0&&rear==-1)||(rear!=front-1))
                        q[++rear]=x;
        }
}

void delete(){
        int a;
        if((front==0)&&(rear==-1)){
                printf("Queue is underflow !\n");
                exit(1);
        }
        if(front==rear){
                a=q[front];
                rear=-1;
                front=0;
        }else if(front==max-1){
                a=q[front];
                front=0;
        }else
                a=q[front++];

        printf("Deleted element is : %d\n",a);
}

void display(){
        int i,j;
        if(front==0&&rear==-1){
                printf("Queue is underflow !\n");
                exit(1);
        }
        if(front>rear){
                for(i=0;i<=rear;i++)
                        printf("%d ",q[i]);
                for(j=front;j<=max-1;j++)
                        printf("%d ",q[j]);
        }else{
                for(i=front;i<=rear;i++)
                        printf("%d ",q[i]);
        }

        printf("\nRear is at %d",q[rear]);
        printf("\nFront is at %d\n",q[front]);
}

void main(){
        int ch;

        printf("\nCircular Queue operations\n");
        printf("1. Insert\n2. Delete\n3. Display\n4. Exit\n");

        while(1){
                printf("\nEnter your choice : ");
                scanf("%d",&ch);

                switch(ch){
                        case 1:
                                insert();
                                break;
                        case 2:
                                delete();
                                break;
                        case 3:
                                display();
                                break;
                        case 4:
                                exit(1);
                        default:
                                printf("Invalid option !\n");
                }
        }
}
```

7. Design, Develop and Implement a menu driven Program in C for the following operations on 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

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

struct node{
        char usn[20],name[10],branch[5];
        unsigned long long int phno;
        int sem;
        struct node *link;
};

typedef struct node *NODE;
NODE temp,FIRST=NULL;

NODE getnode(){
        NODE x;
        x=(NODE)malloc(sizeof(struct node));
        x->link=NULL;
        return x;
}

void read(){
        temp=getnode();
        printf("Enter USN : ");
        scanf("%s",temp->usn);
        printf("Enter the name :\n");
        scanf("%s",temp->name);
        printf("Enter Branch : ");
        scanf("%s",temp->branch);
        printf("Enter the phone number :\n");
        scanf("%llu",&temp->phno);
        printf("Enter the semester : ");
        scanf("%d",&temp->sem);
}

void insert_front(){
        read();
        if(FIRST==NULL)
                FIRST=temp;
        else{
                temp->link=FIRST;
                FIRST=temp;
        }
} void create_SSL(){
        int n,i;
        printf("Enter the number of students : ");
        scanf("%d",&n);
        for(i=1;i<=n;i++){
                printf("Enter the details of the student %d\n",i);
                insert_front();
        }
}

void display_count(){
        int count=1;
        temp=FIRST;
        printf("Student details :\n");
        if(FIRST==NULL)
                printf("Student detail is NULL and the count is 0\n");
        else{
                printf("\nUSN\tNAME\t\tBRANCH\t\tPHONE\t\tSEMESTER\n");
                while(temp->link!=NULL){
                        count++;
                        printf( \
                                "%s\t%s\t\t%s\t\t%llu\t\t%d\n", \
                                temp->usn,temp->name,temp->branch,temp->phno,temp->sem \
                        );
                        temp=temp->link;
                }
                printf("\nStudent count is %d\n",count);
        }
}

NODE insert_end(){
        NODE last=FIRST;
        printf("Enter the details of the students :\n");
        read();
        if(FIRST==NULL)
                FIRST=temp;
        else{
                while(last->link!=NULL)
                        last=last->link;
                last->link=temp;
        }
        return FIRST;
}

void delete_front(){
        temp=FIRST;
        if(FIRST==NULL)
                printf("List is empty !\n");
        else{
                printf("Deleted item is %s\n",temp->usn);
                FIRST=FIRST->link;
                free(temp);
        }
}

void delete_end(){
        NODE last=NULL;
        temp=FIRST;
        if(FIRST==NULL)
                printf("List is empty !\n");
        else if(FIRST->link==NULL){
                printf("Deleted item is %s\n",temp->usn);
                free(FIRST);
                FIRST=NULL;
        }else{
                while(temp->link!=NULL){
                        last=temp;
                        temp=temp->link;
                }
                last->link=NULL;
                printf("Deleted element is %s\n",temp->usn);
                free(temp);
        }
}

void main(){
        int choice;
                printf( \
                        "\n1.Create SSL   \n2.Display SSL  \n3.Insert front \n4.Insert end \
                         \n5.Delete front \n6.Delete End   \n7.Exit" \
                );
        while(1){
                printf("\n> ");
                scanf("%d",&choice);
                switch(choice){
                        case 1:
                                create_SSL();
                                break;
                        case 2:
                                display_count();
                                break;
                        case 3:
                                printf("Enter the details of the students\n");
                                insert_front();
                                break;
                        case 4:
                                insert_end();
                                break;
                        case 5:
                                delete_front();
                                break;
                        case 6:
                                delete_end();
                                break;
                        case 7:
                                return;
                        default:
                                printf("Invalid choice !\n");
                }
        }
}
```

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

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

int n=0,count=0;
struct node{
        char ssn[12],name[20],dept[25],desig[20];
        unsigned long long int phno;
        float sal;
        struct node *prev,*next;
};

typedef struct node *NODE;
NODE x,temp,FIRST=NULL,END=NULL;

NODE getnode(){
        x=(NODE)malloc(sizeof(struct node));
        x->prev=x->next=NULL;
        return x;
}

void read(){
        temp=getnode();
        printf("Enter SSN : ");
        scanf("%s",temp->ssn);
        printf("Enter Name :\n");
        scanf("%s",temp->name);
        printf("Enter Department : ");
        scanf("%s",temp->dept);
        printf("Enter Designation : ");
        scanf("%s",temp->desig);
        printf("Enter Phone : ");
        scanf("%llu",&temp->phno);
        printf("Enter Salary : ");
        scanf("%f",&temp->sal);
}

void Insertionend(){
        printf("Enter the details of the employee\n");
        read();
        if(FIRST==NULL)
                FIRST=END=temp;
        else{
                END->next=temp;
                temp->prev=END;
                END=temp;
        }
}

void Create_DLL(){
        printf("Enter the number of Employees : ");
        scanf("%d",&n);
        while(n>0)
                Insertionend();
}

void display_count(){
        temp=FIRST;
        if(FIRST==NULL)
                printf("The Employee detail is NULL and count is %d\n",count);
        else{
                printf("Employee details:\n");
                printf("SSN \tEMPLOYEE NAME\tDEPARTMENT\tDESIGNATION\tPHONE NUMBER\tSALARY");
                while(temp!=NULL){
                        count++;
                        printf("\n%s\t%s\t%s\t\t%s\t\t%llu\t\t%0.2f",\
                                temp->ssn,temp->name,temp->dept,temp->desig,temp->phno,temp->sal);
                        temp=temp->next;
                }
                        printf("\n Employee count is %d\n",count);
        }
}

void Insertionfront(){
        printf("Enter the details of the employee\n");
        read();
        if(FIRST==NULL)
                FIRST=END=temp;
        else{
                temp->next=FIRST;
                FIRST->prev=temp;
                FIRST=temp;
        }
}

void Deletionfront(){
        temp=FIRST;
        if(FIRST==NULL)
                printf("List is empty !\n");
        else{
                printf("Deleted element is %s\n",temp->ssn);
                count--;
                if(FIRST==END)
                        FIRST=END=NULL;
                else{
                        FIRST=FIRST->next;
                        FIRST->prev=NULL;
                }
                free(temp);
        }
}

void Deletionend(){
        temp=END;
        if(FIRST==NULL)
                printf("List is empty !\n");
        else{
                printf("Deleted element is %s\n",temp->ssn);
                count--;
                if(FIRST==END)
                        FIRST=END=NULL;
                else{
                        END=END->prev;
                        END->next=NULL;
                }
                free(temp);
        }
}

int main(){
        int choice;
                printf("\n1.Create DLL of N Employees \
                        \n2.Display DLL \
                        \n3.Insertion at front \
                        \n4.Insertion at end \
                        \n5.Deletion at front \
                        \n6.Deletion at end \
                        \n7.Exit\n"
                );
        while(1){
                printf("\n> ");
                scanf("%d",&choice);
                        switch(choice){
                                case 1 :
                                        Create_DLL();
                                        break;
                                case 2:
                                        display_count();
                                        break;
                                case 3:
                                        Insertionfront();
                                        break;
                                case 4:
                                        Insertionend();
                                        break;
                                case 5:
                                        Deletionfront();
                                        break;
                                case 6:
                                        Deletionend();
                                        break;
                                case 7:
                                        exit(0);
                                        break;
                                default: printf("Invalid Choice !\n");
                        }
        }
}
```

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

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

   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

→
```c
#include<stdio.h>
#include<stdlib.h>
#include<math.h>

struct node{
        int coef,x,y,z;
        struct node *link;
};

typedef struct node *NODE;
NODE temp,head,cur,x,a=NULL,b,c;

NODE getnode(){
        x=(NODE)malloc(sizeof(struct node));
        return x;
}

NODE readpoly(){
        char ch;
        head=getnode();
        head->coef=head->x=head->y=head->z=-1;
        head->link=head;
        loop:
        temp=getnode();
        printf("\nEnter the coefficient and exponents in decreasing order : ");
        scanf("%d%d%d%d",&temp->coef,&temp->x,&temp->y,&temp->z);
        cur=head;
        while(cur->link!=head)
                cur=cur->link;
        cur->link=temp;
        temp->link=head;
        printf("\nDo you want to enter more coefficients (Y/N) : ");
        fflush(stdin);
        scanf(" %c",&ch);
        if(ch=='y'||ch=='Y')
                goto loop;
        return head;
}

int compare(NODE a,NODE b){
        if((a->x>b->x)||(a->y>b->y)||(a->z>b->z))
                return 1;
        else if((a->x<b->x)||(a->y<b->y)||(a->z<b->z))
                return -1;
        return 0;
}

void attach(int cf,int x1,int y1,int z1,NODE *ptr){
        temp=getnode();
        temp->coef=cf;
        temp->x=x1;
        temp->y=y1;
        temp->z=z1;
        (*ptr)->link=temp;
        *ptr=temp;
}

NODE addpoly(NODE a,NODE b){
        NODE starta;
        int sum,done=0;
        starta=a;
        a=a->link;
        b=b->link;
        c=getnode();
        c->coef=c->x=c->y=c->z=-1;
        NODE lastc=c;
        do{
                switch(compare(a,b)){
                        case -1:
                                attach(b->coef,b->x,b->y,b->z,&lastc);
                                b=b->link;
                                break;
                        case 0:
                                if(starta==a)
                                        done=1;
                                else{
                                        sum=a->coef+b->coef;
                                        if(sum)
                                                attach(sum,a->x,a->y,a->z,&lastc);
                                        a=a->link;
                                        b=b->link;
                                }
                                break;
                        case 1:
                                if(starta==a)
                                        done=1;
                                attach(a->coef,a->x,a->y,a->z,&lastc);
                                a=a->link;
                                break;
                }
        }while(!done);
        lastc->link=c;
        return c;
}

void print(NODE ptr){
        cur=ptr->link;
        while(cur!=ptr){
                printf("%d*x^%d*y^%d*z^%d",cur->coef,cur->x,cur->y,cur->z);
                cur=cur->link;
                if(cur!=ptr)
                printf(" + ");
        }
}

void evaluate(NODE ptr){
        int res=0,x,y,z,ex,ey,ez,cof;

        printf("\nEnter the values of x,y,z : ");
        scanf("%d%d%d",&x,&y,&z);

        cur=ptr->link;

        while(cur!=ptr){
                ex=cur->x;
                ey=cur->y;
                ez=cur->z;
                cof=cur->coef;
                res+=cof*pow(x,ex)*pow(y,ey)*pow(z,ez);

                cur=cur->link;
        }
        printf("\nresult: %d",res);
}

void main(){
        int ch;
        printf("\n1. Represent first polynomial A \
                \n2. Represent Second polynomial B \
                \n3. Display the polynomial A \
                \n4. Display the polynomial B \
                \n5. Add A & B polynomials \
                \n6. Evaluate polynomial C \
                \n7. Exit"
        );

        while(1){
                printf("\n> ");
                scanf("%d",&ch);
                switch(ch){
                        case 1:
                                printf("\nEnter the elements of the polynomial A");
                                a=readpoly();
                                break;
                        case 2:
                                printf("\nEnter the elements of the polynomial B");
                                b=readpoly();
                                break;
                        case 3:
                                print(a);
                                break;
                        case 4:
                                print(b);
                                break;
                        case 5:
                                c=addpoly(a,b);
                                printf("\nThe sum of two polynomials is : ");
                                print(c);
                                printf("\n");
                                break;
                        case 6:
                                evaluate(c);
                                break;
                        case 7:
                                return;
                        default:
                                printf("\nInvalid choice !\n");
                }
        }
}
```

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 In-order, Pre-order and Post-order
    c. Search the BST for a given element (KEY) and report the appropriate message
    d. Exit

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

struct node{
        int value;
        struct node *ltree,*rtree;
};

typedef struct node *NODE;

NODE getnode(){
        NODE x;
        x=(NODE)malloc(sizeof(struct node));
        x->ltree=x->rtree=NULL;
        return x;
}

NODE create(int item,NODE root){
        NODE temp,cur,prev;
        temp=getnode();
        temp->value=item;
        if(root==NULL){
                root=temp;
                return root;
        }
        prev=NULL;
        cur=root;
        while (cur!=NULL){
                prev=cur;
                if(temp->value==cur->value)
                        return root;
                cur=(temp->value<cur->value)?cur->ltree:cur->rtree;
        }
        if(temp->value<prev->value)
                prev->ltree=temp;
        else if(temp->value>prev->value)
                prev->rtree=temp;
        return root;
}

void in(NODE IN){
        if(IN!=NULL){
                in(IN->ltree);
                printf("%d\t",IN->value);
                in(IN->rtree);
        }
}

void pre(NODE PRE){
        if(PRE!=NULL){
                printf("%d\t",PRE->value);
                pre(PRE->ltree);
                pre(PRE->rtree);
        }
}

void post(NODE POST){
        if(POST!=NULL){
                post(POST->ltree);
                post(POST->rtree);
                printf("%d\t",POST->value);
        }
}

void search(NODE root){
        int item;
        NODE cur;
        printf("Enter the element to be searched : ");
        scanf("%d",&item);
        if(root==NULL)
                printf("tree is empty\n");
        cur=root;
        while(cur!=NULL){
                if(item==cur->value){
                        printf("Found key %d in tree\n",cur->value);
                        return ;
                }
                if(item<cur->value)
                        cur=cur->ltree;
                else
                        cur=cur->rtree;
        }
        printf("Key not found\n");
}

int main(){
        int choice,item,n,i;
        NODE root=NULL;
        printf( \
                "\n1. Create BST of N Integers \
                 \n2. BST Traversal \
                 \n3. Binary Search \
                 \n4. Exit"
        );
        while(1){
                printf("\n> ");
                scanf("%d",&choice);
                switch(choice){
                        case 1:
                                printf("\nEnter number elements : ");
                                scanf("%d",&n);
                                printf("Enter the item(s) to be inserted : \n");

                                for(i=0;i<n;i++){
                                        scanf("%d",&item);
                                        root=create(item,root);
                                }

                                break;
                        case 2:

                                if(root==NULL)
                                        printf("Tree is empty\n");
                                else{

                                        printf("\n\nPREORDER traversal\n");
                                        pre(root);
                                        printf("\n\nINORDER traversal\n");
                                        in(root);
                                        printf("\n\nPOSTORDER traversal\n");
                                        post(root);
                                }
                                break;
                        case 3:
                                search(root);
                                break;
                        case 4:
                                exit(0);
                        default:
                                printf("\nInvalid Choice !\n");
                }
        }
}
```

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 given staring node in diagrams using DFS, BFS methods

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

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

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

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

int main(){
        int ch,start;
        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]);
                visited[i]=0;
        }

        printf("\nEnter the starting vertex: ");
        scanf("%d",&start);
        printf( \
                "\n1. BFS : Print all nodes reachable from a given starting node \
                 \n2. DFS : Print all nodes reachable from a given starting node \
                 \n3. Exit \
                 \n> "
        );
        scanf("%d",&ch);

        switch(ch){
                case 1:
                        printf("\nNodes reachable from starting vertex %d are :\n",start);
                        bfs(start);
                        for(i=1;i<=n;i++)
                                if(visited[i]==0)
                                        printf("\nThe vertex that is not reachable is %d\n",i);
                        break;
                case 2:
                        printf("\nNodes reachable from starting vertex %d are :\n",start);
                        dfs(start);
                        break;
                case 3:
                        exit(0);
                default:
                        printf("Please enter valid choice !\n");
        }
}
```

12. Given a file of N employees with a set of K keys (4 digits) which uniquely determine the records in file F. Assume that file F is maintained in memory by Hash table (HT) of m memory locations with L on the set of memory address in L are integers
Develop a program in C that uses :

a . hash function H:K $\rightarrow$ L as H(K) = K mod m (remainder method)

b . Implement hashing technique to map a given key K to the address space L

c . Resolve the collision (if any) using linear probing

$\rightarrow$

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

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

void insert(int key){
        ind=key%m;
        while(ht[ind]!=-1)
                ind=(ind+1)%m;
        ht[ind]=key;
        count++;
}

void display(){
        if(count==0){
                printf("\nHash Table is empty !\n");
                exit(0);
        }

        printf("\nHash Table contents are :\n");
        for(i=0;i<m;i++)
                printf("\n T[%d] --> %d ",i,ht[i]);
        printf("\n");
        printf("Total records Inserted : %d\n",count);
}

void main(){
        printf("\nEnter the number of employee records (N) : ");
        scanf("%d",&n);

        printf("\nEnter the two digit memory locations (m) for hash table : ");
        scanf("%d",&m);
        ht=(int *)malloc(m*sizeof(int));

        for(i=0;i<m;i++)
        ht[i]=-1;

        printf("\nEnter 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("\nHash table is full ! Cannot insert the record %d key",i+1);
                        break;
                }
                insert(key[i]);
        }
        display();
}
```