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.

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

#define NUM_DAYS_IN_WEEK 7

// Structure to represent a day
typedef struct
{
    char *acDayName;    // Dynamically allocated string for the day name
    int iDate;          // Date of the day
    char *acActivity;   // Dynamically allocated string for the activity description
}DAYTYPE;

void fnFreeCal(DAYTYPE *);
void fnDispCal(DAYTYPE *);
void fnReadCal(DAYTYPE *);
DAYTYPE *fnCreateCal();

int main()
```

```
{
   // Create the calendar
   DAYTYPE *weeklyCalendar = fnCreateCal();

   // Read data from the keyboard
   fnReadCal(weeklyCalendar);

   // Display the week's activity details
   fnDispCal(weeklyCalendar);

   // Free allocated memory
   fnFreeCal(weeklyCalendar);

   return 0;
}

DAYTYPE *fnCreateCal()
{
   DAYTYPE *calendar = (DAYTYPE *)malloc(NUM_DAYS_IN_WEEK * sizeof(DAYTYPE));



   for(int i = 0; i < NUM_DAYS_IN_WEEK; i++)
           {
      calendar[i].acDayName = NULL;
      calendar[i].iDate = 0;
      calendar[i].acActivity = NULL;
   }

   return calendar;
}
```

```
void fnReadCal(DAYTYPE *calendar)
{
        char cChoice;
  for(int i = 0; i < NUM_DAYS_IN_WEEK; i++)
        {
    printf("Do you want to enter details for day %d [Y/N]: ", i + 1);
    scanf("%c", &cChoice); getchar();

    if(tolower(cChoice) == 'n')
        continue;

    printf("Day Name: ");
    char nameBuffer[50];
    scanf("%s", nameBuffer);
    calendar[i].acDayName = strdup(nameBuffer);  // Dynamically allocate and copy the string



    printf("Date: ");
    scanf("%d", &calendar[i].iDate);



    printf("Activity: ");
    char activityBuffer[100];
    scanf(" %[^\n]", activityBuffer);  // Read the entire line, including spaces
    calendar[i].acActivity = strdup(activityBuffer);

    printf("\n");
    getchar();          //remove trailing enter character in input buffer
  }
}
```

```
void fnDispCal(DAYTYPE *calendar)
{
   printf("\nWeek's Activity Details:\n");
   for(int i = 0; i < NUM_DAYS_IN_WEEK; i++)
          {
      printf("Day %d:\n", i + 1);
                    if(calendar[i].iDate == 0)
                    {
                              printf("No Activity\n\n");
                              continue;
                    }

      printf("  Day Name: %s\n", calendar[i].acDayName);
      printf("  Date: %d\n", calendar[i].iDate);
      printf("  Activity: %s\n\n", calendar[i].acActivity);
   }
}

void fnFreeCal(DAYTYPE *calendar)
{
   for(int i = 0; i < NUM_DAYS_IN_WEEK; i++)
          {
      free(calendar[i].acDayName);
      free(calendar[i].acActivity);
   }
   free(calendar);
}
```

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

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

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>
void main()
{
  char s[20],pat[20],rep[20],ans[30];
  int i,j,k,l,flag;
  printf("\nEnter string:");
  scanf("%s",s);
  printf("\nEnter pattern:");
  scanf("%s",pat);
  printf("\nEnter replacement:");
  scanf("%s",rep);
  for(i=0,k=0;s[i]!='\0';i++)
  {
    flag=1;
    for(j=0;pat[j]!='\0';j++)
      if(s[i+j]!=pat[j])
        flag=0;
    l=j;
    if(flag)
    {
      for(j=0;rep[j]!='\0';j++,k++)
        ans[k]=rep[j];
      i+=l-1;
    }
    else
      ans[k++]=s[i];
  }
  ans[k]='\0';
  printf("%s",ans);


}
```

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).

Push an Element on to Stack

Pop an Element from Stack

Demonstrate how Stack can be used to check Palindrome

Demonstrate Overflow and Underflow situations on Stack

Display the status of Stack

Exit

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

```
#include <stdio.h>
#include <stdlib.h>
int s[5],top=-1;

void push()
{
   if(top==4)
      printf("\nStack overflow!!!!");
   else
   {
      printf("\nEnter element to insert:");
      scanf("%d",&s[++top]);
   }
}

void pop()
{
   if(top==-1)
      printf("\nStack underflow!!!");
   else
      printf("\nElement popped is: %d",s[top--]);
}
void disp()
{
   int t=top;
```

```
    if(t==-1)
        printf("\nStack empty!!");
    else
        printf("\nStack elements are:\n");
    while(t>=0)
        printf("%d ",s[t--]);
}
void pali()
{
    int num[5],rev[5],i,t;
    for(i=0,t=top;t>=0;i++,t--)
        num[i]=rev[t]=s[t];
    for(i=0;i<=top;i++)
        if(num[i]!=rev[i])
        break;
    /*printf(" num     rev\n");
    for(t=0;t<=top;t++)
      printf("%4d   %4d\n",num[t],rev[t]);*///remove /*  */ to display num and rev
    if(i==top+1)
        printf("\nIt is a palindrome");
    else
        printf("\nIt is not a palindrome");
}
int main()
{
    int ch;
    do
    {
        printf("\n...Stack operations.....\n");
        printf("1.PUSH\n");
        printf("2.POP\n");
        printf("3.Palindrome\n");
        printf("4.Display\n");
        printf("5.Exit\n________________\n");
        printf("Enter choice:");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:push();break;
            case 2:pop();break;
            case 3:pali();break;
            case 4:disp();break;
            case 5:exit(0);
            default:printf("\nInvalid choice");
        }
    }
    while(1);
    return 0;

}
```

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

```
#include<stdio.h>
#include<string.h>
int F(char symbol)
{
 switch (symbol)
 {
 case '+':
 case '-':return 2;
 case '*':
 case '/':
 case '%':return 4;
 case '^':
 case '$':return 5;
 case '(':return 0;
 case '#':return -1;
 default :return 8;
 }
}
int G(char symbol)
{
 switch (symbol)
 {
 case '+':
 case '-':return 1;
 case '*':
 case '/':
 case '%':return 3;
 case '^':
 case '$':return 6;
 case '(':return 3;
 case ')':return 0;
 default :return 7;
 }
}
void infix_postfix(char infix[], char postfix[])
{
int top=-1, j=0, i;
char s[30], symbol;
s[++top] = '#';
for(i=0; i < strlen(infix); i++)
{
 symbol = infix[i];
 while (F(s[top]) > G(symbol))
 {
  postfix[j] = s[top--];
  j++;
 }
 if(F(s[top]) != G(symbol))
  s[++top] = symbol;
 else
  top--;
}
while(s[top] != '#')
 postfix[j++] = s[top--];
postfix[j] = '\0';
}
void main()
{
char infix[20], postfix[20];
printf("\nEnter a valid infix expression\n") ;
scanf ("%s", infix) ;
infix_postfix (infix, postfix);
```

```
printf("\nThe infix expression is:\n");
printf ("%s",infix);
printf("\nThe postfix expression is:\n");
printf ("%s",postfix) ;
}
```

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

```
#include<stdio.h>
#include<math.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 '$':
  case '^': return pow(op1,op2);
  default : return 0;
 }
}
void main()
{
float s[20], res, op1, op2;
int top, i;
char postfix[20], symbol;
printf("\nEnter 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;
 }
}
res = s[top--];
printf("\nThe result is : %f\n", res);
}

5b. #include <stdio.h>
void towers(int, char, char, char);
int main()
{
    int num;
    printf("Enter the number of disks : ");
    scanf("%d", &num);
    printf("The sequence of moves involved in the Tower of Hanoi are :n");
    towers(num, 'A', 'C', 'B');
    printf("n");
    return 0;
}
void towers(int num, char frompeg, char topeg, char auxpeg)
{
    if (num == 1)
    {
        printf("n Move disk 1 from peg %c to peg %c", frompeg, topeg);
        return;
    }
    towers(num - 1, frompeg, auxpeg, topeg);

    printf("n Move disk %d from peg %c to peg %c", num, frompeg, topeg);

    towers(num - 1, auxpeg, topeg, frompeg);

}
```

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)

1. Insert an Element on to Circular QUEUE
2. Delete an Element from Circular QUEUE
3. Demonstrate Overflow and Underflow situations on Circular QUEUE
4. Display the status of Circular QUEUE
5. Exit

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

```
#include <stdio.h>
#include <stdlib.h>
#define max 5
int q[max],f=-1,r=-1;
void ins()
{
   if(f==(r+1)%max)
      printf("\nQueue overflow");
   else
   {
      if(f==-1)
         f++;
      r=(r+1)%max;
      printf("\nEnter element to be inserted:");
      scanf("%d",&q[r]);
   }
}
void del()
{
   if(r==-1)
      printf("\nQueue underflow");
   else
   {
      printf("\nElemnt deleted is:%d",q[f]);
      if(f==r)
         f=r=-1;
      else
         f=(f+1)%max;
   }
}
void disp()
{
   if(f==-1)
      printf("\nQueue empty");
   else
   {
        int i;
        printf("\nQueue elements are:\n");
        for(i=f;i!=r;i=(i+1)%max)
            printf("%d\t",q[i]);
        printf("%d",q[i]);
        printf("\nFront is at:%d\nRear is at:%d",q[f],q[r]);
    }
}
int main()
{
    printf("\nCircular Queue operations");
    printf("\n1.Insert");
    printf("\n2.Delete");
    printf("\n3.Display");
    printf("\n4.Exit");
    int ch;
    do{
        printf("\nEnter choice:");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:ins();break;
            case 2:del();break;
            case 3:disp();break;
            case 4:exit(0);
            default:printf("\nInvalid choice...!");
        }
   }while(1);
    return 0;
}
```

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

1. Create a SLL of N Students Data by using front insertion.
2. Display the status of SLL and count the number of nodes in it
3. Perform Insertion / Deletion at End of SLL
4. Perform Insertion / Deletion at Front of SLL(Demonstration of stack)
5. Exit

```
#include<string.h>
#include<stdio.h>
#include<stdlib.h>
struct stud
{
   char usn[11],name[15],branch[4],phno[11];
   int sem;
   struct stud *next;
}*f=NULL,*r=NULL,*t=NULL;
void ins(int ch)
{
   t=(struct stud*)malloc(sizeof(struct stud));
   printf("\nEnter USN:");
   scanf("%s",t->usn);
   printf("Enter Name:");
   scanf("%s",t->name);
   printf("Enter Branch:");
   scanf("%s",t->branch);
   printf("Enter Sem:");
   scanf("%d",&t->sem);
   printf("Enter Phno:");
   scanf("%s",t->phno);
   t->next=NULL;
   if(!r)
      f=r=t;
   else
   {
      if(ch)
      {
         r->next=t;
         r=t;
      }
      else
      {
         t->next=f;
         f=t;
```

```
        }
    }
}
void del(int ch)
{
    if(!f)
        printf("\nList Empty");
    else
    {
        struct stud *t1;
        if(f==r)
        {
            t1=f;
            f=r=NULL;
        }
        else if(ch)
        {
            t1=r;
            for(t=f;t->next!=r;t=t->next)
                r=t;
            r->next=NULL;
        }
        else
        {
            t1=f;
            f=f->next;
        }
        printf("\nElement deleted is:\n");
        printf("USN:%s\nName:%s\nBranch:%s\nSem:%d\nPhno:%s\n",t1->usn,t1->name,t1-
>branch,t1->sem,t1->phno);
        free(t1);
    }
}
void disp()
{
    if(!f)
        printf("\nList Empty!!!");
    else
        printf("\nList elements are:\n");
    for(t=f;t;t=t->next)
        printf("\nUSN:%s\nName:%s\nBranch:%s\nSem:%d\nPhno:%s\n",t->usn,t->name,t->branch,t-
>sem,t->phno);
}
void main()
{
    int ch,n,i;
    printf("\n........Menu..........,\n");
    printf("1.Create\n");
    printf("2.Display\n");
    printf("3.Insert at end\n");
    printf("4.Delete at end\n");
    printf("5.Insert at beg\n");
    printf("6.Delete at beg\n");
    printf("7.Exit\n");
    while(1)
    {
        printf("\nEnter choice:");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1: printf("\nEnter no. of nodes:");
                    scanf("%d",&n);
                    for(i=0;i<n;i++)
                        ins(0);
                    break;
            case 2:disp();break;
            case 3:ins(1);break;
            case 4:del(1);break;
            case 5:ins(0);break;
            case 6:del(0);break;
            case 7:exit(0);
            default:printf("\nInvalid choice!!!!");
        }
    }

}
```

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

1. Create a DLL of N Employees Data by using end insertion.
2. Display the status of DLL and count the number of nodes in it
3. Perform Insertion and Deletion at End of DLL
4. Perform Insertion and Deletion at Front of DLL
5. Demonstrate how this DLL can be used as Double Ended Queue
6. Exit

```
#include<string.h>
int count=0;
struct node
{
struct node *prev;
int ssn,phno;
float sal;
char name[20],dept[10],desg[20];
struct node *next;
}*h,*temp,*temp1,*temp2,*temp4;
void create()
{
int ssn,phno;
float sal;
char name[20],dept[10],desg[20];
 temp =(struct node *)malloc(sizeof(struct node));
 temp->prev = NULL;
 temp->next = NULL;
 printf("\n Enter ssn,name,department, designation, salary and phno of employee : ");
 scanf("%d %s %s %s %f %d", &ssn, name,dept,desg,&sal, &phno);
 temp->ssn = ssn;
 strcpy(temp->name,name);
 strcpy(temp->dept,dept);
 strcpy(temp->desg,desg);
 temp->sal = sal;
 temp->phno = phno;
 count++;
}
void insertbeg()
{
if (h == NULL)
 {
create();
 h = temp;
 temp1 = h;
 }
```

```
else
 {
 create();
 temp->next = h;
 h->prev = temp;
 h = temp;
 }
}
void insertend()
{
if(h==NULL)
 {
 create();
 h = temp;
 temp1 = h;
 }
else
 {
 create();
 temp1->next = temp;
 temp->prev = temp1;
 temp1 = temp;
 }
}
void displaybeg()
{
 temp2 =h;
if(temp2 == NULL)
 {
 printf("List empty to display \n");
 return;
 }
 printf("\n Linked list elements from begining : \n");
while (temp2!= NULL)
 {
 printf("%d %s %s %s %f %d\n", temp2->ssn, temp2->name,temp2->dept,
 temp2->desg,temp2->sal, temp2->phno );
 temp2 = temp2->next;
}
 printf(" No of employees = %d ", count);
}
int deleteend()
{
struct node *temp;
 temp=h;
```

```
if(temp->next==NULL)
 {
 free(temp);
 h=NULL;
 return 0;
 }
else
 {
 temp2=temp1->prev;
 temp2->next=NULL;
 printf("%d %s %s %s %f %d\n", temp1->ssn, temp1->name,temp1->dept,
 temp1->desg,temp1->sal, temp1->phno );
 free(temp1);
 }
 count--;
return 0;
}
int deletebeg()
{
struct node *temp;
 temp=h;
if(temp->next==NULL)
 {
 free(temp);
 h=NULL;
 }
else
 {
 h=h->next;
 printf("%d %s %s %s %f %d", temp->ssn, temp->name,temp->dept,
 temp->desg,temp->sal, temp->phno );
 free(temp);
 }
 count--;
return 0;
}
void main()
{
int ch,n,i;
 h=NULL;
 temp = temp1 = NULL;
 printf("-----------------MENU--------------------\n");
 printf("\n 1 - create a DLL of n emp");
 printf("\n 2 - Display from beginning");
 printf("\n 3 - Insert at end");
 printf("\n 4 - delete at end");
 printf("\n 5 - Insert at beg");
 printf("\n 6 - delete at beg");
 printf("\n 7 - exit\n");
 printf("------------------------------------------\n");
while (1)
 {
 printf("\n Enter choice : ");
 scanf("%d", &ch);
 switch (ch)
 {
 case 1:
 printf("\n Enter no of employees : ");
 scanf("%d", &n);
 for(i=0;i<n;i++)
 insertend();
 break;
 case 2:
 displaybeg();
 break;
 case 3:
 insertend();
 break;
 case 4:
 deleteend();
 break;
 case 5:
 insertbeg();
 break;
 case 6:
 deletebeg();
 break;
 case 7:
 exit(0);
 default:
printf("wrong choice\n");
 }
 }
}
```

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

1. Represent and Evaluate a Polynomial P(x,y,z) = 6x2y2z-4yz5+3x3yz+2xy5z-2xyz3
2. 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<math.h>
#include<stdlib.h>
#include<math.h>
typedef struct node
{
int expo,coef;
struct node *next;
}node;
/*FUNCTION PROTOTYPE*/
node * insert(node *,int,int);
node * create();
node * add(node *p1,node *p2);
int eval(node *p1);
void display(node *head);
node *insert(node*head,int expo1,int coef1)
{
 node *p,*q;
 p=(node *)malloc(sizeof(node));
 p->expo=expo1;
 p->coef=coef1;
 p->next=NULL;
if(head==NULL)
 {
 head=p;
 head->next=head;
 return(head);
 }

if(expo1>head->expo)
 {
 p->next=head->next;
 head->next=p;
 head=p;
 return(head);
 }
if(expo1==head->expo)
 {
```

```
 head->coef=head->coef+coef1;
return(head);
 }
 q=head;
while(q->next!=head&&expo1>=q->next->expo)
 q=q->next;
if(p->expo==q->expo)
 q->coef=q->coef+coef1;
else
 {
 p->next=q->next;
 q->next=p;
 }
return(head);
}
node *create()
{
int n,i,expo1,coef1;
 node *head=NULL;
 printf("\n\nEnter no of terms of polynomial==>");
 scanf("%d",&n);
for(i=0;i<n;i++)
 {
 printf("\n\nEnter coef & expo==>");
 scanf("%d%d",&coef1,&expo1);
 head=insert(head,expo1,coef1);
 }
return(head);
}
node *add(node *p1,node *p2)
{
 node *p;
 node *head=NULL;
 printf("\n\n\nAddition of polynomial==>");
 p=p1->next;
do
 {
 head=insert(head,p->expo,p->coef);
 p=p->next;
 }while(p!=p1->next);
 p=p2->next;
do
 {
head=insert(head,p->expo,p->coef);
 p=p->next;
 }while(p!=p2->next);
return(head);
}
int eval(node *head)
{
 node *p;
int x,ans=0;
 printf("\n\nEnter the value of x=");
 scanf("%d",&x);
 p=head->next;
do
 {
 ans=ans+p->coef*pow(x,p->expo);
 p=p->next;
 }while(p!=head->next);
return(ans);
}
void display(node *head)
{
 node *p,*q;
int n=0;
 q=head->next;
 p=head->next;
do
 {
 n++;
 q=q->next;
 }while(q!=head->next);
 printf("\n\n\tThe polynomial is==>");

do
 {
 if(n-1)
 {
 printf("%dx^(%d) + ",p->coef,p->expo);
 p=p->next;
 }
 else
 {
printf(" %dx^(%d)",p->coef,p->expo);
 p=p->next;
 }
 n--;
 } while(p!=head->next);
}
```

```
void main()
{
int a,x,ch;
 node *p1,*p2,*p3;
 p1=p2=p3=NULL;
while(1)
 {
 printf("\n\t----------------<< MENU >>---------------");
 printf("\n\tPolynomial Operations :");
 printf(" 1.Add");
 printf("\n\t\t\t\t\t2.Evaluate");
 printf("\n\t\t\t\t\t3.Exit");
 printf("\n\t------------------------------------------- ");
 printf("\n\n\n\tEnter your choice==>");
 scanf("%d",&ch);
 switch(ch)
 {
 case 1 :
 p1=create();
 display(p1);
 p2=create();
 display(p2);
 p3=add(p1,p2);
 display(p3);
 break;
 case 2 :
 p1=create();
 display(p1);
 a=eval(p1);
 printf("\n\nValue of polynomial=%d",a);
 break;
 case 3 :
 exit(0);
 break;
 default :
 printf("\n\n\t invalid choice");
 break;
 }
}
}
```

<u>OUTPUT</u>

```
POLY1(x, y, z) = 6x^2y^2z^1 + 4x^0y^1z^5 + 3x^3y^1z^1 + 2x^1y^5z^1 + 2x^1y^1z^3
POLY2(x, y, z) = 1x^1y^1z^1 + 4x^3y^1z^1
POLYSUM(x, y, z) = 6x^2y^2z^1 + 4x^0y^1z^5 + 7x^3y^1z^1 + 2x^1y^5z^1 + 2x^1y^1z^3 +
1x^1y^1z^1      Result of POLYSUM(1, 2, 3): 2364
```

10. Design, Develop and Implement a menu driven Program in C for the following operations on Binary Search Tree(BST) of Integers

1. Create a BST of N Integers: 6, 9, 5, 2, 8, 15, 24, 14, 7, 8, 5, 2
2. Traverse the BST in Inorder, Preorder and Post Order
3. Search the BST for a given element (KEY) and report the appropriate message
4. Exit

```
#include <stdio.h>
#include <stdlib.h>
int flag=0;
typedef struct BST
{
int data;
struct BST *lchild,*rchild;
} node;
/*FUNCTION PROTOTYPE*/
void insert(node *, node *);
void inorder(node *);
void preorder(node *);
void postorder(node *);
node *search(node *, int, node **);
void main()
{
int choice;
int ans =1;
int key;
 node *new_node, *root, *tmp, *parent;
 node *get_node();
 root = NULL;
 printf("\nProgram For Binary Search Tree ");
do
 {
 printf("\n1.Create");
 printf("\n2.Search");
 printf("\n3.Recursive Traversals");
 printf("\n4.Exit");
 printf("\nEnter your choice :");
 scanf("%d", &choice);
 switch (choice)
 {
 case 1:
 do
 {
 new_node = get_node();
 printf("\nEnter The Element ");
 scanf("%d", &new_node->data);
if (root == NULL) /* Tree is not Created */
 root = new_node;
 else
 insert(root, new_node);
 printf("\nWant To enter More Elements?(1/0)");
 scanf("%d",&ans);
 } while (ans);
 break;
 case 2:
 printf("\nEnter Element to be searched :");
 scanf("%d", &key);
 tmp = search(root, key, &parent);
 if(flag==1)
 {
 printf("\nParent of node %d is %d", tmp->data, parent->data);
 }
 else
 {
 printf("\n The %d Element is not Present",key);
 }
 flag=0;
 break;
 case 3:
 if (root == NULL)
 printf("Tree Is Not Created");
 else
 {
 printf("\nThe Inorder display :");
 inorder(root);
 printf("\nThe Preorder display : ");
 preorder(root);
 printf("\nThe Postorder display : ");
 postorder(root);
 }
 break;
 }
 }
 while (choice != 4);
 }
/*Get new Node */
node *get_node()
{
 node *temp;
```

```
temp = (node *) malloc(sizeof(node));
 temp->lchild = NULL;
 temp->rchild = NULL;
return temp;
}
/*This function is for creating a binary search tree */
void insert(node *root, node *new_node)
{
if (new_node->data < root->data)
 {
 if(root->lchild==NULL)
 root->lchild=new_node;
 else
 insert(root->lchild, new_node);
 }
if (new_node->data > root->data)
 {
 if (root->rchild == NULL)
 root->rchild = new_node;
 else
 insert(root->rchild, new_node);
 }
}
/*This function is for searching the node from binary Search Tree*/
node *search(node *root, int key, node **parent)
{
 node *temp;
 temp = root;
while (temp != NULL)
 {
 if (temp->data == key)
 {
 printf("\nThe %d Element is Present", temp->data);
 flag=1;
 return temp;
 }
 *parent = temp;
 if (temp->data > key)
 temp = temp->lchild;
 else
 temp = temp->rchild;
 }
return NULL;
}
/*This function displays the tree in inorder fashion */
```

```
void inorder(node *temp)
{
if (temp != NULL)
 {
 inorder(temp->lchild);
 printf("%d\t", temp->data);
 inorder(temp->rchild);
 }
}
/*This function displays the tree in preorder fashion */
void preorder(node *temp)
{
if (temp != NULL)
 {
 printf("%d\t", temp->data);
 preorder(temp->lchild);
 preorder(temp->rchild);
 }
}

/*This function displays the tree in postorder fashion */
void postorder(node *temp)
{
if (temp != NULL)
 {
 postorder(temp->lchild);
 postorder(temp->rchild);
 printf("%d\t", temp->data);
 }
}
```

<u>OUTPUT</u>

```
Create a BST of N Integers
Enter the number N : 6

Enter 6 numbers

50

30

70

40

60
```

```
20
1.Inorder traversal
2.Preorder traversal
3.Postorder traversal
4.Search
5.Exit

Enter your choice : 1
Inorder Traversal is :
20      30      40      50      60      70




1.Inorder traversal
2.Preorder traversal
3.Postorder traversal
4.Search
5.Exit
Enter your choice : 2
Preorder Traversal is :
50      30      20      40      70      60

1.Inorder traversal
2.Preorder traversal
3.Postorder traversal
4.Search
```

```
5.Exit


Enter your choice : 3
Postorder Traversal is :
20      40      30      60      70      50
1.Inorder traversal
2.Preorder traversal
3.Postorder traversal
4.Search
5.Exit
Enter your choice : 4
Enter the element to be searched : 30
30 is found in the BST
1.Inorder traversal
2.Preorder traversal
3.Postorder traversal
4.Search
5.Exit
Enter your choice : 4
Enter the element to be searched : 80
80 is not found in the BST
1.Inorder traversal
2.Preorder traversal
3.Postorder traversal
4.Search
5.Exit
Enter your choice : 5
```

11. Design, Develop and Implement a Program in C for the following operations on Graph(G) of Cities
    1. Create a Graph of N cities using Adjacency Matrix.
    2. 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[20][20],q[20],visited[20],reach[10],n,i,j,f=0,r= -1,count=0;
void bfs(int v)
{
for(i=1;i<=n;i++)
 if(a[v][i] && !visited[i])
 q[++r]=i;
if(f<=r)
 {
 visited[q[f]]=1;
 bfs(q[f++]);
 }
}
void dfs(int v)
{
int i;
 reach[v]=1;
for(i=1;i<=n;i++)
 {
 if(a[v][i] && !reach[i])
 {
 printf("\n %d->%d",v,i);
 count++;
 dfs(i);
 }
 }
}
void main()
{
int v, choice;
 printf("\n Enter the number of vertices:");
 scanf("%d",&n);
for(i=1;i<=n;i++)
 {
 q[i]=0;
 visited[i]=0;
 }
for(i=1;i<=n-1;i++)
 reach[i]=0;
printf("\n Enter graph data in matrix form:\n");
for(i=1;i<=n;i++)
 for(j=1;j<=n;j++)
 scanf("%d",&a[i][j]);
 printf("1.BFS\n 2.DFS\n 3.Exit\n");
 scanf("%d",&choice);
switch(choice)
 {
 case 1:
 printf("\n Enter the starting vertex:");
 scanf("%d",&v);
 bfs(v);
 if((v<1)||(v>n))
 {
 printf("\n Bfs is not possible");
 }
 else
 {
 printf("\n The nodes which are reachable from %d:\n",v);
 for(i=1;i<=n;i++)
 if(visited[i])
 printf("%d\t",i);
 }
 break;
 case 2:
 dfs(1);
 if(count==n-1)
 printf("\n Graph is connected");
 else
 printf("\n Graph is not connected");
 break;
 case 3:
 exit(0);
 }
}
```

OUTPUT

```
Enter the number of vertices : 4

Enter the adjacency matrix :

0 1 0 0
```

```
1 0 0 0
0 0 0 1
0 0 1 0
Enter the starting vertex : 1
Vertices which can be reached from vertex 1 are :-
1 2

putta:~/…/Programs$ ./a.out
Enter the number of vertices : 4
Enter the adjacency matrix :
0 1 1 0
1 0 0 1
1 0 0 1
0 1 1 0
Enter the starting vertex : 1
Vertices which can be reached from vertex 1 are :-
1 2 3 4
putta:~/…/Programs$ ./a.out
Enter the number of vertices : 4
Enter the adjacency matrix :
0 1 0 0
1 0 0 0
0 0 0 1
0 0 1 0
Enter the starting vertex : 3
Vertices which can be reached from vertex 3 are :-
3 4
```

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>

#define MAX 100

/*FUNCTION PROTOTYPE */

int create(int);

void linear_prob(int[], int, int);

void display (int[]);

void main()

{

int a[MAX],num,key,i;

int ans=1;

 printf(" collision handling by linear probing : \n");

for (i=0;i<MAX;i++)

 {

 a[i] = -1;

 }

do

 {

 printf("\n Enter the data");

 scanf("%4d", &num);

 key=create(num);

 linear_prob(a,key,num);

 printf("\n Do you wish to continue ? (1/0) ");

 scanf("%d",&ans);
 }while(ans);
 display(a);
}
int create(int num)
{
int key;
 key=num%100;
return key;
}
void linear_prob(int a[MAX], int key, int num)
{
int flag, i, count=0;
 flag=0;
if(a[key]== -1)
{
 a[key] = num;
 }
else
 {
 printf("\nCollision Detected...!!!\n");
 i=0;
 while(i<MAX)
 {
 if (a[i]!=-1)
 count++;
 i++;
 }
 printf("Collision avoided successfully using LINEAR PROBING\n");
 if(count == MAX)
 {
 printf("\n Hash table is full");
 display(a);
 exit(1);
 }
 for(i=key+1; i<MAX; i++)
 if(a[i] == -1)
 {
 a[i] = num;
 flag =1;
 break;
 }
 //for(i=0;i<key;i++)
 i=0;
 while((i<key) && (flag==0))
 {
 if(a[i] == -1)
 {
 a[i] = num;
 flag=1;
 break;
 }
 i++;
 }
 }
}
void display(int a[MAX])
{
int i,choice;
```

```
 printf("1.Display ALL\n 2.Filtered Display\n");
 scanf("%d",&choice);
if(choice==1)
 {
 printf("\n the hash table is\n");
 for(i=0; i<MAX; i++)
 printf("\n %d %d ", i, a[i]);
 }
else
 {
 printf("\n the hash table is\n");
 for(i=0; i<MAX; i++)
 if(a[i]!=-1)
 {
 printf("\n %d %d ", i, a[i]);
 continue;
 }
 }
}
```

OUTPUT

```
Enter the size of the hash table (m): 50
Enter a key to search for an employee record: 5216
Employee found with key 5216:
Name: sanjay

putta:~/.../Programs$ ./a.out
```

```
Enter the size of the hash table (m): 50

Enter a key to search for an employee record: 4327

Employee found with key 4327:

Name: nahar
```