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 z4yz 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 poly
{
    int cf,px,py,pz;
    int flag;
    struct poly *link;
};
typedef struct poly *NODE;
NODE insertrear(NODE h,int cf,int px,int py,int pz)
{
    NODE temp,cur;
    temp=(NODE)malloc(sizeof(struct poly));
    temp->cf=cf;
    temp->px=px;
    temp->py=py;
    temp->pz=pz;
    if(h->link==h)
    {
        h->link=temp;
```

```
                temp->link=h;
                return temp;
        }
        cur=h->link;
        while(cur->link!=h)
        cur=cur->link;
        temp->link=h;
        return h;
}
void readpoly(NODE h)
{
        int cf,px,py,pz,ch;
        do
        {
                printf("Enter the coefficient px,py,pz:\n");
                scanf("%d%d%d%d",&cf,&px,&py,&pz);
                h=insertrear(h,cf,px,py,pz);
                printf("To continue 1, to exit 0: ");
                scanf("%d",&ch);
                }while(ch);
                return;
}
void evalpoly(NODE h1)
{
	int x,y,z;
	float result=0.0;
	NODE temp=h1->link;
	printf("Enter the values of x,y,z:\n");
	scanf("%d%d%d",&x,&y,&z);
	while(temp!=h1)
	{
		result=result+temp->cf*pow(x,temp->px)*pow(y,temp->py)*pow(z,temp->pz);
		temp=temp->link;
	}
	printf("The result: %f",result);
}
void display(NODE h1)
{
	NODE temp;
	if(h1->link==h1)
	{
		printf("Polynomial empty\n");
		return;
	}
	temp=h1->link;
	while(temp!=h1)
	{
		if(temp->cf>0)
```

```
                printf("+%dx^%d y^%d z^%d",temp->cf,temp->px,temp->py,temp->pz);
                else
                printf("%dx^%d y^%d z^%d",temp->cf,temp->px,temp->py,temp->pz);
                temp=temp->link;
        }
}
void polyadd(NODE h1,NODE h2,NODE h3)
{
        NODE p1,p2;
        int cf1,cf,cf2,px1,px2,py1,py2,pz1,pz2;
        p1=h1->link;
        while(p1!=h2)
        {
                cf1=p1->cf;
                px1=p1->px;
                py1=p1->py;
                pz1=p1->pz;
                p1=p1->link;
        while(p2!=h2)
        {
                cf2=p2->cf;
                px2=p2->px;
                py2=p2->py;
                pz2=p2->pz;
                if(px1==px2&&py1==py2&&pz1==pz2)
                break;
                p2=p2->link;
        }
        if(p2!=h2)
        {
                cf=cf1+cf2;
                p1->flag=1;
                if(cf!=0)
                h3=insertrear(h3,cf,px1,py1,pz1);
                p1=p1->link;
                p2=p2->link;
        }
        else
        {
                h3=insertrear(h3,cf1,px1,py1,pz1);
                p1=p1->link;
        }
}
p2=h2->link;
while(p2!=h2)
{
        if(p2->flag==0)
        insertrear(h3,p2->cf,p2->px,p2->py,p2->pz);
        p2=p2->link;
}
return ;
}
int main()
{
        int ch;
        NODE h1,h2,h3;
        h1=(NODE)malloc(sizeof(struct poly));
        h2=(NODE)malloc(sizeof(struct poly));
        h3=(NODE)malloc(sizeof(struct poly));
        h1=h1->link;
        h2=h2->link;
        h3=h3->link;
        while(1)
        {
                printf("1.Evaluate a polynomial\n2.Add polynomial\n3.Exit\n");
                printf("Enter the choice:\n");
                scanf("%d",&ch);
                switch(ch)
                {
                        case 1:printf("Enter the polynomial:\n");
                        readpoly(h1);
                        evalpoly(h1);
```

```
                    display(h1);
                    h1->link=h1;
                    break;
                    case 2:printf("Enter the first polynomial:\n");
                    readpoly(h1);
                    printf("Enter the 2nd poly:\n");
                    readpoly(h2);
                    polyadd(h1,h2,h3);
                    printf("1st polynomial is:\n");
                    display(h1);
                    printf("2nd polynomial is:\n");
                    display(h2);
                    printf("\n");
                    printf("Resultant polynomial is:\n");
                    display(h3);
                    printf("\n");
                    break;
                    case 3:exit(0);
                }
        }
}
```

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 *left;
	struct BST *right;
};
typedef struct BST NODE;
NODE *node;
NODE* createtree(NODE *node, int data)
{
	if(node==NULL)
	{
		NODE *temp;
		temp=(NODE*)malloc(sizeof(NODE));
		temp->data=data;
		temp->left=temp->right=NULL;
		return temp;
	}
	if(data<node->data)
```

```
        node->left=createtree(node->left,data);
        else if(data>node->data)
        node->right=createtree(node->right,data);
        return node;
}
NODE* search(NODE *node,int data)
{
        if(node==NULL)
        printf("Element not found\n");
        else if(data<node->data)
        search(node->left,data);
        else if(data>node->data)
        search(node->right,data);
        else
        printf("Element found is: %d",node->data);
        return node;
}
void inorder(NODE *node)
{
        if(node!=NULL)
        {
                inorder(node->left);
                printf("%d\t",node->data);
                inorder(node->right);
        }
}
void preorder(NODE *node)
{
        if(node!=NULL)
        {
                printf("%d\t",node->data);
                preorder(node->left);
                preorder(node->right);
        }
}
void postorder(NODE *node)
{
        if(node!=NULL)
        {
                preorder(node->left);
                preorder(node->right);
                printf("%d\t",node->data);
        }
}
NODE* findmin(NODE *node)
{
        if(node==NULL)
        return NULL;
        if(node->left)
        return findmin(node->left);
        else
        return node;
}
NODE* del(NODE *node,int data)
{
        NODE *temp;
        if(node==NULL)
        printf("Element not found\n");
        else if(data<node->data)
        node->left=del(node->left,data);
        else if(data>node->data)
        node->right=del(node->right,data);
        else
        {
                if(node->right&&node->left)
                {
                        temp=findmin(node->right);
                        node->data=temp->data;
                        node->right=del(node->right,temp->data);
                 }
                 else
                 {
```

```
                temp=node;
                if(node->left==NULL)
                node=node->right;
                else if(node->right==NULL)
                node=node->left;
                free(temp);
            }
    }
    return node;
}
int main()
{
    int data,ch,i,n;
    NODE *root=NULL;
    while(1)
    {
        printf("1.Insertion\,\n2.Search\n3.Delete element\n4.IN order\n5.Preorder\n6.postorder\n7.Exit\n");
        printf("Enter you choice:\n");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:printf("Enter the value:\n");
            scanf("%d",&n);
            printf("Enter the values to create BST tree");
    for(i=0;i<n;i++)

    {

        scanf("%d",&data);

        root=createtree(root,data);

    }

    break;

    case 2:printf("Enter the element to search: ");

    scanf("%d",&data);

    break;

    case 3:printf("Enter the element to delete:\n");

    scanf("%d",&data);

    root=del(root,data);

    break;

    case 4:printf("Inorder:\n");

    inorder(root);

    break;

    case 5:printf("Preorder:\n");

    preorder(root);

    break;

    case 6:printf("Post order:\n");

    postorder(root);

    break;

    case 7:exit(0);

}
```

```
        }
}
```

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[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);
                }
        }
}
int main()
{
        int v,ch;
        printf("Enter the number vertices:\n");
        scanf("%d",&n);
        for(i=1;i<=n;i++)
        {
                q[i]=0;
                visited[i]=0;
        }
        for(i=1;i<=n;i++)
        reach[i]=0;
        printf("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\n2.DFS\n3.Exit\n");
printf("Enter your choice:\n");
scanf("%d",&ch);
switch(ch)
{
        case 1:printf("Enter the starting vertex:");
        scanf("%d",&v);
        bfs(v);
        if((v<1)||(v>n))
        printf("BFS not found\n");
        else
        {
                printf("The nodes which are reachable from %d:\n");
                for(i=1;i<=n;i++)
                if(visited[i])
                printf("%d\t",i);
        }
        break;
        case 2:dfs(1);
        if(count==n-1)
        printf("Graph is connected\n");
        else
        printf("Graph not connected\n");
        break;
```

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

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>
#define MAX 100
int create(int);
void display(int[]);
void linear_prob(int[],int,int);
int 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("Enter the data:\n");
                scanf("%4d",&num);

                key=create(num);

                linear_prob(a,key,num);

                printf("\nDo you want to continue? (0/1): ");

                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("Collinsion 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("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 ch,i;
    printf("1.Display All\n2.Filtered display\n");
    scanf("%d",&ch);
    if(ch==1)
    {
        printf("\nThe hash table is:\n");
        for(i=0;i<MAX;i++)
        printf("%d %d\n",i,a[i]);
    }
    else
    {
        printf("The hash table is\n");
        for(i=0;i<MAX;i++)
        if(a[i]!=-1)
        {
            printf("%d %d\n ",i,a[i]);
            continue;
        }
    }
}
```