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

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

```
#include<stdio.h>
#include<stdio.h>
#include<malloc.h>
#include<stdlib.h>
#include<math.h>
struct node
{
        int cf, px, py, pz;
int flag;
        struct node *link;
};
typedef struct node NODE;

NODE* getnode()
{
        NODE *x;
        x=(NODE*)malloc(sizeof(NODE));
        if(x==NULL)
        {
                printf("Insufficient memory\n");
                exit(0);
        }
        return x;
}

void display(NODE *head)
{

        NODE *temp;
        if(head->link==head)
        {
                printf("Polynomial does not exist\n");
                return;
        }
        temp=head->link;
        printf("\n");
        while(temp!=head)
        {
                printf("%d x^%d y^%d z^%d",temp->cf,temp->px,temp->py,temp->pz);
                if(temp->link != head)
                printf(" + ");
                temp=temp->link;
```

```
        }
        printf("\n");
}
NODE* insert_rear(int cf,int x,int y,int z,NODE *head)
{

        NODE *temp,*cur;
        temp=getnode();
        temp->cf=cf;
        temp->px=x;
        temp->py=y;
        temp->pz=z;
        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 px, py, pz, cf;
        int ch;
        printf("\nEnter coeff: ");
        scanf("%d",&cf);
        printf("\nEnter x, y, z powers(0-indiacate NO term): "); scanf("%d%d%d", &px, &py, &pz);
        head=insert_rear(cf,px,py,pz,head);
        printf("\nIf you wish to continue press 1 otherwise 0: ");
        scanf("%d",&ch);
        while(ch!=0)
        {
                printf("\nEnter coeff: ");
                scanf("%d",&cf);
                printf("\nEnter x, y, z powers(0-indiacate NO term): ");
                scanf("%d%d%d", &px, &py, &pz); head=insert_rear(cf,px,py,pz,head);
                printf("\nIf you wish to continue press 1 otherwise 0: ");
                scanf("%d", &ch);
        }
        return head;
}
NODE* add_poly(NODE *h1,NODE *h2,NODE *h3)
{
        NODE *p1,*p2;
        int x1,x2,y1,y2,z1,z2,cf1,cf2,cf;
```

```
	p1=h1->link;
	while(p1!=h1)
	{
		x1=p1->px;
		y1=p1->py;
		z1=p1->pz;
		cf1=p1->cf;
		p2=h2->link;
		while(p2!=h2)
		{
			x2=p2->px;
			y2=p2->py;
			z2=p2->pz;
			cf2=p2->cf;
			if(x1==x2 && y1==y2 && z1==z2)
			break;
			p2=p2->link;
		}
		if(p2!=h2)
		{
			cf=cf1+cf2;
			p2->flag=1;
			if(cf!=0)
			h3=insert_rear(cf,x1,y1,z1,h3);
		}
		else
			h3=insert_rear(cf1,x1,y1,z1,h3);
			p1=p1->link;
	}
	p2=h2->link;
	while(p2!=h2)
	{
		if(p2->flag==0)
		h3=insert_rear(p2->cf,p2->px,p2->py,p2->pz,h3);
		p2=p2->link;
	}
	return h3;
}
void evaluate(NODE *he)
{
	NODE *head;
	int x, y, z;
	float result=0.0;
	head=he;
	printf("\nEnter x, y, z, terms to evaluate:\n");
	scanf("%d%d%d", &x, &y, &z);
	he=he->link;
	while(he != head)
```

```
	{
		result = result + (he->cf * pow(x,he->px) * pow(y,he->py) * pow(z,he->pz));
		he=he->link;
	}
	printf("\nPolynomial result is: %f", result);
}
void main()
{

	NODE *h1,*h2,*h3,*he;
	int ch;
	while(1)
	{
		printf("\n\n1.Evaluate polynomial\n2.Add two polynomials\n3.Exit\n");
		printf("Enter your choice: ");
		scanf("%d", &ch);
		switch(ch)
		{
			case 1:  he=getnode();
				he->link=he;
printf("\nEnter polynomial to evaluate:\n");
				he=read_poly(he);
				display(he);
				evaluate(he);
				free(he);
				break;
			case 2:  h1=getnode();
				h2=getnode();
	h3=getnode();
	h1->link=h1;
	h2->link=h2;
	h3->link=h3;
printf("\nEnter the first polynomial:");
				h1=read_poly(h1);
				printf("\nEnter the second polynomial:");
				h2=read_poly(h2);
				h3=add_poly(h1,h2,h3);
				printf("\nFirst polynomial is: ");
				display(h1);
				printf("\nSecond polynomial is: ");
				display(h2);
				printf("\nThe sum of 2 polynomials is: "); display(h3);
				break;
			case 3:exit(0);
				break;
			default:printf("\nInvalid entry");
				break;
		}
```

```
}
}
```
**OUTPUT:-**

```
1.Evaluate polynomial
2.Add two polynomials
3.Exit
Enter your choice: 1
Enter polynomial to evaluate:

Enter coeff: 6
Enter x, y, z powers(0-indiacate NO term):
2 2 1
If you wish to continue press 1 otherwise 0: 1
Enter coeff: -4
Enter x, y, z powers(0-indiacate NO term):
0 1 5
If you wish to continue press 1 otherwise 0: 1
Enter coeff: 3
Enter x, y, z powers(0-indiacate NO term):
3 1 1
If you wish to continue press 1 otherwise 0: 1
Enter coeff: 2
Enter x, y, z powers(0-indiacate NO term):
1 5 1
If you wish to continue press 1 otherwise 0: 1
Enter coeff: -2
Enter x, y, z powers(0-indiacate NO term):
1 1 3
If you wish to continue press 1 otherwise 0: 0
6 x^2 y^2 z^1 + -4 x^0 y^1 z^5 + 3 x^3 y^1 z^1 + 2 x^1 y^5 z^1 + -2 x^1 y^1 z^3
Enter x, y, z, terms to evaluate:
2 2 2
Polynomial result is: 224.000000

1.Evaluate polynomial
2.Add two polynomials
3.Exit
Enter your choice: 2
Enter the first polynomial:
Enter coeff: 1
Enter x, y, z powers(0-indiacate NO term):
1 1 1
If you wish to continue press 1 otherwise 0: 1
Enter coeff: 3
Enter x, y, z powers(0-indiacate NO term):
3 4 5
If you wish to continue press 1 otherwise 0: 0
```

```
Enter the second polynomial:
Enter coeff: 2
Enter x, y, z powers(0-indiacate NO term):
5 4 6
If you wish to continue press 1 otherwise 0: 1
Enter coeff: 1
Enter x, y, z powers(0-indiacate NO term):
1 1 1
If you wish to continue press 1 otherwise 0: 0
First polynomial is:
1 x^1 y^1 z^1 + 3 x^3 y^4 z^5
Second polynomial is:
2 x^5 y^4 z^6 + 1 x^1 y^1 z^1
The sum of 2 polynomials is:
2 x^1 y^1 z^1 + 3 x^3 y^4 z^5 + 2 x^5 y^4 z^6
1.Evaluate polynomial
2.Add two polynomials
3.Exit
Enter your choice: 3
```

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

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

struct BST
{
  int item;
  struct BST *llink, *rlink;
};
typedef struct BST* NODE;

NODE insert(NODE);
void inorder(NODE);
void preorder(NODE);
void postorder(NODE);
NODE search(NODE, int);
NODE Delete(NODE, int);

void main()
{
  int choice, key,n,i;
```

```
  NODE root = NULL, tmp, parent;
while(1)
  {
    printf("\n1.Create");
printf("\n2.Traverse the Tree in Preorder, Inorder, Postorder");
    printf("\n3.Search");
printf("\n4.Delete an element from the Tree");
    printf("\n5.Exit");
printf("\nEnter your choice :");
scanf("%d", &choice);

    switch (choice)
    {
          case 1:   printf("\n enter the number of nodes");
                     scanf("%d",&n);
                     for(i=0;i<n;i++)
                    root = insert(root);
                    break;

          case 2: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;
          case 3:printf("\nEnter Element to be searched :");
                    scanf("%d", &key);
                     tmp = search(root, key);
                    if(tmp == NULL)
                          printf("Element does not exists\n");
                     else
                          printf("\nThe element %d  found", tmp->item);
                     break;
          case 4: printf("\nEnter Element to be deleted :");
                    scanf("%d", &key);
                     root = Delete(root, key);
                     break;
          default: exit(0);
    }
  }
}
/* This function is for creating a binary search tree */
```

```
NODE insert(NODE root)
{
  NODE temp, cur, prev;
  int item;
printf("\nEnter The Element ");
scanf("%d", &item);
  temp = (NODE) malloc(sizeof(struct BST));
  temp->llink = NULL;
  temp->rlink = NULL;
  temp->item = item;

  if (root == NULL)
          return temp;
  prev = NULL;
  cur = root;
while(cur != NULL)
  {
    prev = cur;
    if (item < cur-> item)
          cur = cur->llink;
    else
          cur = cur->rlink;
  }
  if (item < prev->item)
          prev->llink = temp;
  else
          prev->rlink = temp;
  return root;
}

/*  This function displays the tree in inorder fashion */
void inorder(NODE root)
{
  if (root != NULL)
  {
    inorder(root->llink);
printf("%d\t", root->item);
    inorder(root->rlink);
  }
}
/* This function displays the tree in preorder fashion */
void preorder(NODE root)
{
  if (root != NULL)
  {
printf("%d\t", root->item);
    preorder(root->llink);
    preorder(root->rlink);
```

```
  }
}
/* This function displays the tree in postorder fashion  */
void postorder(NODE root)
{
  if (root != NULL)
  {
    postorder(root->llink);
    postorder(root->rlink);
printf("%d\t", root->item);
  }
}
NODE search(NODE root, int key)
{
        NODE cur;
        if(root == NULL)
                return NULL;
        cur = root;
        while(cur != NULL)
        {
                if(key == cur->item)
                        return cur;
                if(key<cur->item)
                        cur = cur->llink;
                else
                        cur = cur->rlink;
        }
        return NULL;
}
NODE Delete(NODE root, int data)
{
        NODE temp;
        int min;

        if (root == NULL)
        {
            return NULL;
        }
        if (data < root->item)
        {  // data is in the left sub tree.
                root->llink = Delete(root->llink, data);
        }
        else if (data > root->item)
        { // data is in the right sub tree.
                root->rlink = Delete(root->rlink, data);
        }
        else
        {
```

```
            // case 1: no children
            if (root->llink == NULL && root->rlink == NULL)
                  {
                  free(root); // wipe out the memory, in C, use free function
                        root = NULL;
                  }
            // case 2: one child (right)
            else if (root->llink == NULL)
                  {
                        temp = root; // save current node as a backup
                        root = root->rlink;
                        free(temp);
                  }
            // case 3: one child (left)
            else if (root->rlink == NULL)
                  {
                        temp = root; // save current node as a backup
                        root = root->llink;
                        free(temp);
            }
            // case 4: two children
            else
                  {
                        min = FindMin(root->rlink); // find minimal value of right sub tree
                        root->item = min; // duplicate the node
                        root->rlink = Delete(root->rlink, min); // delete the duplicate node
            }
          }
          return root; // parent node can update reference
}
int FindMin(NODE root) {
  if (root == NULL) {
    return -1; // or undefined.
  }
  if (root->llink != NULL) {
    return FindMin(root->llink); // left tree is smaller
  }
  return root->item;
}
```

**OUTPUT:-**

```
1.Create
2.Traverse the Tree in Preorder, Inorder, Postorder
3.Search
4.Delete an element from the Tree
5.Exit
Enter your choice :1
 enter the number of nodes 7
Enter The Element 16
Enter The Element 9
Enter The Element 11
Enter The Element 24
Enter The Element 27
Enter The Element 4
Enter The Element 17
1.Create
2.Traverse the Tree in Preorder, Inorder, Postorder
3.Search
4.Delete an element from the Tree
5.Exit
Enter your choice :2
The Inorder display : 4   9        11       16       17       24       27
The Preorder display : 16        9        4        11       24       17       27
The Postorder display : 4        11       9        17       27       24       16
1.Create
2.Traverse the Tree in Preorder, Inorder, Postorder
3.Search
4.Delete an element from the Tree
5.Exit
Enter your choice :3
Enter Element to be searched :11
The element 11  found
1.Create
2.Traverse the Tree in Preorder, Inorder, Postorder
3.Search
4.Delete an element from the Tree
5.Exit
Enter your choice :3
Enter Element to be searched :22
Element does not exists

1.Create
2.Traverse the Tree in Preorder, Inorder, Postorder
3.Search
4.Delete an element from the Tree
5.Exit
Enter your choice :4
Enter Element to be deleted :16
```

1.Create
2.Traverse the Tree in Preorder, Inorder, Postorder
3.Search
4.Delete an element from the Tree
5.Exit
Enter your choice : 2
The Inorder display : 4 9 11 17 24 27
The Preorder display : 17 9 4 11 24 27
The Postorder display : 4 11 9 27 24 17
1.Create
2.Traverse the Tree in Preorder, Inorder, Postorder
3.Search
4.Delete an element from the Tree
5.Exit
Enter your choice :5