****

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

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

****

```
#include<stdio.h>
#include<stdlib.h>
struct SLL
{
  char usn[11];
  char name[30];
  char branch[4];
  int sem;
  char phno[11];
  struct SLL *link;
};
typedef struct SLL node;
node *start=NULL;
node *getnode()
{
  node *newnode;
  newnode=(node*)malloc(sizeof(node));
  if(newnode==NULL)
          printf("Memory overflow");
  else
     {
          printf("\n Enter USN, name, branch, sem, phonenumber\n");
          scanf("%s%s%s%d%s",newnode->usn,newnode->name,newnode->branch,
                          &newnode->sem,newnode->phno);
          newnode->link=NULL;
          return newnode;
  }
}
void insert_front()
{
  node *nn;
  nn=getnode();
  nn->link=start;
  start=nn;
}
```

```
void delete_front()
{
  node *temp;
  if(start==NULL)
          printf("\n List is empty");
  else
  {
  temp=start;
  start=start->link;
  free(temp);
  }
}
void insert_end()
{
  node *nn,*temp;

  nn=getnode();
  if(start==NULL)
          start=nn;
  else
  {
  temp=start;
  while(temp->link!=NULL)
          temp=temp->link;
  temp->link=nn;
  }
}
void delete_end()
{
  node *temp=start,*prev;
  if(start==NULL)
  {
          printf("\n Empty list");
          return;
  }
  if(start->link==NULL)
          start=NULL;
  else
  {
  while(temp->link!=NULL)
  {
          prev=temp;
          temp=temp->link;
  }
  prev->link=NULL;
  }
```

```
  free(temp);
}
void display()
{
  int c=0;
  node *temp;
  if(start==NULL)
  printf("\n Empty list");
  else
  {
  temp=start;
  printf("\n The details are");
  while(temp!=NULL)
  {
          printf("\n%s\t%s\t%s\t%d\t%s\t",temp->usn,temp->name,temp->branch,
          temp->sem,temp->phno);
          c++;
          temp=temp->link;
  }
  printf("\n Number of nodes is %d",c);
  }
}
void stackDemoUsingSLL()
{
        int ch;
        while(1)
        {
                printf("\n~~~Stack Demo using SLL~~~\n");
                printf("\n1:Push operation \n2: Pop operation \n3: Display \n4:Exit \n");
                printf("\nEnter your choice for stack demo");
                scanf("%d", &ch);
                switch(ch)
                {
                        case 1:     insert_front();
                                    break;
                        case 2:     delete_front();break;

                        case 3:     display();
                                    break;
                        default : return;
                }
                 return;
              }

}
```

```
void queueDemoUsingSLL()
{
        int ch;
        while(1)
        {
                printf("\n~~~queue Demo using SLL~~~\n");
                printf("\n1:Insert operation \n2: Delete operation \n3: Display \n4:Exit \n");
                printf("\nEnter your choice for stack demo");
                scanf("%d", &ch);
                switch(ch)
                {
                        case 1:     insert_end();
                                    break;
                        case 2:     delete_front();break;

                        case 3:     display();
                                    break;
                        default : return;
                }
                 return;
              }

}
int main()
{
  int n,m,i;
  while(1)        {
  printf("\n Enter manu\n 1:insert_front\n 2:insert_end\n 3:delete_front\n 4:delete_end\n 5:display \n6.Stack
  Demo using SLL \n 7:Queue Demo using SLL\n8:Exit");
  scanf("%d",&m);
  switch(m)
  {
          case 1: printf("\n Enter n:");
                      scanf("%d",&n);
                      for(i=0;i<n;i++)
                            insert_front();
                   break;
          case 2: insert_end();break;
          case 3: delete_front();break;
          case 4: delete_end();break;
          case 5: display();break;
          case 6: stackDemoUsingSLL();break;
          case 7: queueDemoUsingSLL();break;
          default:return 0;
  }}}
```

/***

Program 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, Designation, Department, Salary, Phone no.

(i)Create a DLL of n employee data by using end insertion

(ii)Display the status of DLL and count the number of nodes in it

(iii)Perform insertion and deletion at the front of DLL

(iv) Perform insertion and deletion at the end of DLL

(v)Demonstrate how this DLL can be used as double ended queue

(vi)Exit

****

```
#include<stdio.h>
#include<stdlib.h>
struct DList
{
  int ssn;
  char name[20];
  char desg[20];
  char dept[20];
  float sal;
  long int  phno;
  struct DList *left,*right;
};
typedef struct DList dnode;
dnode *start=NULL;
dnode *getnode()
{
  dnode *newnode;
  newnode=(dnode*)malloc(sizeof(dnode));
  if(newnode==NULL)
          printf("Memory Overflow\n");
  else
  {
          printf("\n Enter ssn, name, designation,department,salary, phone number");
          scanf("%d%s%s%s%f%ld",&newnode->ssn,newnode->name,newnode->desg,newnode->dept,
                  &newnode->sal,&newnode->phno);
          newnode->left=NULL;
          newnode->right=NULL;
  }
  return newnode;
}
void insert_front()
{
  dnode *nn;
  nn=getnode();
  if(start==NULL)
```

```
  start=nn;
  else
  {
  nn->right=start;
  start->left=nn;
  start=nn;
  }
}
void delete_front()
{
  dnode *temp=start;
  if(start==NULL)
  printf("\n List is empty");
  else if(start->right==NULL)//single node
  {
  start=NULL;
  free(temp);
  }
  else
  {
  start=start->right;
  start->left=NULL;
  free(temp);
  }
}
void insert_end()
{
  dnode *nn,*temp;
  nn=getnode();
  if(start==NULL)
  start=nn;
  else
  {
  temp=start;
  while(temp->right!=NULL)
          temp=temp->right;
  temp->right=nn;
  nn->left=temp;
  }
}
void delete_end()
{
  dnode *temp=start;
  if(start==NULL)
  printf("\n Empty list");
  else if(start->right == NULL)
```

```
  {
  start=NULL;
  free(temp);
  }
  else
  {
  while(temp->right != NULL)
          temp=temp->right;
  (temp->left)->right=NULL;
  free(temp);
  }
}
void display()
{
  int c=0;
  dnode *temp=start;
  if(start==NULL)
  printf("\n Empty list");
  else
  {
  printf("\n The details are");
  while(temp!=NULL)
  {
          printf("\n%d\t%s\t%s\t%s\t%.2f\t%ld\t",temp->ssn,temp->name,temp->desg,
                  temp->dept,temp->sal,temp->phno);
          c++;
          temp=temp->right;
  }
  printf("\n Number of nodes is %d",c);
  }
}
int main()
{
  int n,choice,i;
  while(1)
  {
  printf("\n Enter your choice:");
  printf(" \n1:create list\n 2:insert_front\n 3:insert_end\n 4:delete_front\n 5:delete_end\n 6:Display");
  printf("\n Enter your choice:");
  scanf("%d",&choice);
  switch(choice)
  {
          case 1: printf("\n Enter number of nodes to create:");
                          scanf("%d",&n);
                          for(i=0;i<n;i++)
                          insert_front();
```

```
                        break;
            case 2: insert_front();break;
            case 3: insert_end();break;
            case 4: delete_front();break;
            case 5: delete_end();break;
            case 6: display();break;
            default:exit(0);break;
   }
   }
   return 0;
}
```

****

9.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<stdlib.h>
#include<math.h>
#define COMPARE(x, y)      ( (x == y) ? 0 : (x > y) ? 1 : -1)
struct node
{
        int coef;
        int xexp, yexp, zexp;
        struct node *link;
};
typedef struct node *NODE;
NODE getnode()
{
        NODE x;
        x = (NODE) malloc(sizeof(struct node));
        if(x == NULL)
        {
                printf("Running out of memory \n");
                return NULL;
        }
        return x;
}
NODE attach(int coef, int xexp, int yexp, int zexp, NODE head)
{
        NODE temp, cur;
        temp = getnode();
        temp->coef = coef;
        temp->xexp = xexp;
        temp->yexp = yexp;
        temp->zexp = zexp;
        cur = head->link;
        while(cur->link != head)
                cur = cur->link;

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

```
NODE read_poly(NODE head)
{
        int i, j, coef, xexp, yexp, zexp, n;
        printf("\nEnter the no of terms in the polynomial: ");
        scanf("%d", &n);
        for(i=1; i<=n; i++)
        {
                printf("\n\tEnter the %d term: ",i);
                printf("\n\t\tCoef =  ");
                scanf("%d", &coef);
                printf("\n\t\tEnter Pow(x) Pow(y) and Pow(z): ");
                scanf("%d", &xexp);
                scanf("%d", &yexp);
                scanf("%d", &zexp);
                head = attach(coef, xexp, yexp, zexp, head);
        }
         return head;
}

void display(NODE head)
{
        NODE temp;
        if(head->link == head)
        {
                printf("\nPolynomial does not exist.");
                return;
        }
        temp = head->link;

        while(temp != head)
        {
                printf("%dx^%dy^%dz^%d", temp->coef, temp->xexp, temp->yexp, temp->zexp);
                temp = temp->link;
                if(temp != head)
                        printf(" + ");
        }
}
```

```
int poly_evaluate(NODE head)
{
        int x, y, z, sum = 0;
        NODE poly;
        printf("\nEnter the value of x,y and z: ");
        scanf("%d %d %d", &x, &y, &z);
        poly = head->link;
        while(poly != head)
        {
                sum += poly->coef * pow(x,poly->xexp)* pow(y,poly->yexp) * pow(z,poly->zexp);
                poly = poly->link;
        }
        return sum;
}

NODE poly_sum(NODE head1, NODE head2, NODE head3)
{
   NODE a, b;
   int coef;
   a = head1->link;
   b = head2->link;
   while(a!=head1 && b!=head2)
   {
        while(1)
        {
                if(a->xexp == b->xexp && a->yexp == b->yexp && a->zexp == b->zexp)
                {
                        coef = a->coef + b->coef;
                        head3 = attach(coef, a->xexp, a->yexp, a->zexp, head3);
                        a = a->link;
                        b = b->link;
                        break;
                } //if ends here
                if(a->xexp!=0 || b->xexp!=0)
                {
                        switch(COMPARE(a->xexp, b->xexp))
                        {
                         case -1 :  head3 = attach(b->coef, b->xexp, b->yexp, b->zexp, head3);
                                    b = b->link;
                                    break;

                        case 0 :    if(a->yexp > b->yexp)
                                    {
                                        head3 = attach(a->coef, a->xexp, a->yexp, a->zexp, head3);
                                        a = a->link;
                                        break;
```

```
                    }
                    else if(a->yexp < b->yexp)
                    {
                        head3 = attach(b->coef, b->xexp, b->yexp, b->zexp, head3);
                        b = b->link;
                        break;
                    }
                    else if(a->zexp > b->zexp)
                    {
                        head3 = attach(a->coef, a->xexp, a->yexp, a->zexp, head3);
                        a = a->link;
                        break;
                    }
                    else if(a->zexp < b->zexp)
                    {
                        head3 = attach(b->coef, b->xexp, b->yexp, b->zexp, head3);
                        b = b->link;
                        break;
                    }
            case 1 :    head3 = attach(a->coef,a->xexp,a->yexp,a->zexp,head3);
                        a = a->link;
                        break;
            }  //switch ends here
            break;
}  //if ends here
if(a->yexp!=0 || b->yexp!=0)
{
            switch(COMPARE(a->yexp, b->yexp))
            {
                case -1 :   head3 = attach(b->coef, b->xexp, b->yexp, b->zexp, head3);
                            b = b->link;
                            break;
                 case 0 :   if(a->zexp > b->zexp)
                            {
                                head3 = attach(a->coef, a->xexp, a->yexp, a->zexp, head3);
                                a = a->link;
                                break;
                            }
                            else if(a->zexp < b->zexp)
                            {
                                head3 = attach(b->coef, b->xexp, b->yexp, b->zexp, head3);
                                b = b->link;
                                break;
                            }
                  case 1 :  head3 = attach(a->coef, a->xexp, a->yexp, a->zexp, head3);
                            a = a->link;
```

```
                                        break;
                        }
                        break;
                }
                if(a->zexp!=0 || b->zexp!=0)
                {
                        switch(COMPARE(a->zexp,b->zexp))
                        {
                                case -1 :   head3 = attach(b->coef,b->xexp,b->yexp,b->zexp,head3);
                                            b = b->link;
                                            break;
                                case 1 :    head3 = attach(a->coef, a->xexp, a->yexp, a->zexp, head3);
                                            a = a->link;
                                            break;
                        }
                        break;
                }
        }
   }
   while(a!= head1)
   {
         head3 = attach(a->coef,a->xexp,a->yexp,a->zexp,head3);
        a = a->link;
   }
   while(b!= head2)
   {
        head3 = attach(b->coef,b->xexp,b->yexp,b->zexp,head3);
        b = b->link;
   }
   return head3;
}
void main()
{
  NODE head, head1, head2, head3;
  int res, ch;
  head =  getnode();    /* For polynomial evalaution */
  head1 = getnode();    /* To hold POLY1 */
  head2 = getnode();    /* To hold POLY2 */
  head3 = getnode();    /* To hold POLYSUM */

  head->link=head;
  head1->link=head1;
  head2->link=head2;
  head3->link= head3;
```

```
 while(1)
 {
     printf("\n~~~Menu~~~");
     printf("\n1.Represent and Evaluate a Polynomial P(x,y,z)");
     printf("\n2.Find the sum of two polynomials POLY1(x,y,z)");
     printf("\nEnter your choice:");
     scanf("%d",&ch);
     switch(ch)
     {
        case 1:          printf("\n~~~~Polynomial evaluation P(x,y,z)~~~\n");
                         head = read_poly(head);
                         printf("\nRepresentation of Polynomial for evaluation: \n");
                         display(head);
                         res = poly_evaluate(head);
                         printf("\nResult of polynomial evaluation is : %d \n", res);
                         break;

        case 2:          printf("\nEnter the POLY1(x,y,z):  \n");
                         head1 = read_poly(head1);
                         printf("\nPolynomial 1 is:  \n");
                         display(head1);

                         printf("\nEnter the POLY2(x,y,z):  \n");
                         head2 = read_poly(head2);
                         printf("\nPolynomial 2 is: \n");
                         display(head2);

                         printf("\nPolynomial addition result: \n");
                         head3 = poly_sum(head1,head2,head3);
                         display(head3);
                         break;
           case 3:       exit(0);
     }
   }
}
```

Output:

```
~~~Menu~~~
1.Represent and Evaluate a Polynomial P(x,y,z)
2.Find the sum of two polynomials POLY1(x,y,z) and POLY2(x,y,z)
Enter your choice: 1
~~~~Polynomial evaluation P(x,y,z)~~~
Enter the no of terms in the polynomial: 5
      Enter the 1 term:
            Coef =  6
            Enter Pow(x) Pow(y) and Pow(z): 2        2        1
```

Enter the 2 term:
Coef = **-4**
Enter Pow(x) Pow(y) and Pow(z): **0 1 5**
Enter the 3 term:
Coef = **3**
Enter Pow(x) Pow(y) and Pow(z): **3 1 1**
Enter the 4 term:
Coef = **2**
Enter Pow(x) Pow(y) and Pow(z): **1 5 1**
Enter the 5 term:
Coef = **-2**
Enter Pow(x) Pow(y) and Pow(z): **1 1 3**
Representation of Polynomial for evaluation:
6x^2y^2z^1 + -4x^0y^1z^5 + 3x^3y^1z^1 + 2x^1y^5z^1 + -2x^1y^1z^3
Enter the value of x,y and z: **1 1 1**
Result of polynomial evaluation is : **5**

~~~Menu~~~
1.Represent and Evaluate a Polynomial P(x,y,z)
2.Find the sum of two polynomials POLY1(x,y,z) and POLY2(x,y,z)
Enter your choice: **2**

Enter the POLY1(x,y,z):
Enter the no of terms in the polynomial: **5**
Enter the 1 term:
Coef = **6**
Enter Pow(x) Pow(y) and Pow(z): **4 4 4**
Enter the 2 term:
Coef = **3**
Enter Pow(x) Pow(y) and Pow(z): **4 3 1**
Enter the 3 term:
Coef = **5**
Enter Pow(x) Pow(y) and Pow(z): **0 1 1**
Enter the 4 term:
Coef = **10**
Enter Pow(x) Pow(y) and Pow(z): **0 1 0**
Enter the 5 term:
Coef = **5**
Enter Pow(x) Pow(y) and Pow(z): **0 0 0**
Polynomial 1 is:
6x^4y^4z^4 + 3x^4y^3z^1 + 5x^0y^1z^1 + 10x^0y^1z^0 + 5x^0y^0z^0
Enter the POLY2(x,y,z):
Enter the no of terms in the polynomial: **5**
Enter the 1 term:
Coef = **8**
Enter Pow(x) Pow(y) and Pow(z): **4 4 4**
~~~

Enter the 2 term:
Coef = **4**
Enter Pow(x) Pow(y) and Pow(z): **4 2 1**
Enter the 3 term:
Coef = **30**
Enter Pow(x) Pow(y) and Pow(z): **0 1 0**
Enter the 4 term:
Coef = **20**
Enter Pow(x) Pow(y) and Pow(z): **0 0 1**
Enter the 5 term:
Coef = **3**
Enter Pow(x) Pow(y) and Pow(z): **0 0 0**

Polynomial 2 is:
8x^4y^4z^4 + 4x^4y^2z^1 + 30x^0y^1z^0 + 20x^0y^0z^1 + 3x^0y^0z^0

Polynomial addition result:
14x^4y^4z^4 + 3x^4y^3z^1 + 4x^4y^2z^1 + 5x^0y^1z^1 + 40x^0y^1z^0 + 20x^0y^0z^1 + 8x^0y^0z^0

~~~Menu~~~
1.Represent and Evaluate a Polynomial P(x,y,z)
2.Find the sum of two polynomials POLY1(x,y,z) and POLY2(x,y,z)
Enter your choice:3
~~~

****

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 Inorder, Preorder and Post Order

c. Search the BST for a given element (KEY) and report the appropriate message

d. Delete an element(ELEM) from BST

e. Exit

****

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

void insert(NODE root, NODE newnode);
void inorder(NODE root);
void preorder(NODE root);
void postorder(NODE root);
void search(NODE root, int key);

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

```
void search(NODE root, int key)
{
 NODE temp;
 if(root == NULL)
 {
        printf("\nBST is empty.");
        return;
 }
 temp = root;
 while (temp != NULL)
 {
        if (temp->data == key)
         {
                printf("\nKey element %d is present in BST", temp->data);
                return;
        }
         if (key < temp->data)
                 temp = temp->lchild;
         else
                 temp= temp->rchild;
 }
 printf("\nKey element %d is not found in the BST", key);
}
void inorder(NODE root)
{
 if(root != NULL)
 {
 inorder(root->lchild);
 printf("%d ", root->data);
 inorder(root->rchild);
 }
}
void preorder(NODE root)
{
 if (root != NULL)
 {
 printf("%d ", root->data);
 preorder(root->lchild);
 preorder(root->rchild);
 }
}
```

```
void postorder(NODE root)
 {
 if (root != NULL)
 {
 postorder(root->lchild);
 postorder(root->rchild);
 printf("%d ", root->data);
 }}
void main()
{
 int ch, key, val, i, n;
 NODE root = NULL, newnode;
 while(1) {
 printf("\n1.Create a BST(insert)\n2.BST Traversals\n3.Search\n4.Exit");
 printf("\nEnter your choice: ");
 scanf("%d", &ch);
 switch(ch)
 {
          case 1: printf("\nEnter the number of elements: ");
                  scanf("%d", &n);
                  for(i=1;i<=n;i++)
                  {
                        newnode=(NODE)malloc(sizeof(struct BST));
                        printf("\nEnter The value: ");
                        scanf("%d", &val);
                        newnode->data = val;
                        newnode->lchild = NULL;
                        newnode->rchild = NULL;
                        if (root == NULL)
                             root = newnode;
                        else
                            insert(root, newnode);
                   }
                  break;
case 2:           if (root == NULL)
                           printf("\nTree Is Not Created");
                  else
                           {
                           printf("\nThe Preorder display : ");
                           preorder(root);
                           printf("\nThe Inorder display : ");
                           inorder(root);
                           printf("\nThe Postorder display : ");
                           postorder(root);
                           }
                  break;
```

```
 case 3: printf("\nEnter Element to be searched: ");
              scanf("%d", &key);
              search(root, key);
              break;
 case 4: exit(0);
 }
 }
}
```