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>

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

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

NODE getnode(){
        NODE x;
        x=(NODE)malloc(sizeof(struct node));
        x->prev=NULL;
        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 Create_DLL(){
        int n,i=1;
        printf("Enter the number of Employees : ");
        scanf("%d",&n);
        while(i<=n){
                printf("Enter the details of the %d employee\n",i++);
                read();

                if(FIRST==NULL)
                        FIRST=END=temp;
                else{
                        END->next=temp;
                        temp->prev=END;
                        END=temp;
                }
        }
}

void display_count(){
        temp=FIRST;
        int count=0;
        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->s
                        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 Insertionend(){
        printf("Enter the details of the new employee\n");
        read();
        if(FIRST==NULL)
                FIRST=END=temp;
        else{
                END->next=temp;
                temp->prev=END;
                END=temp;
        }

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

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

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

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

typedef struct node *NODE;

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

NODE readpoly(){
        NODE temp,head,cur;
        char ch;
        head=getnode();
        head->coef=head->x=head->y=head->z=-1;
        head->link=head;
        do{
                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);

        }while(ch=='y'||ch=='Y');

        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){
        NODE temp;
        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,c,lastc;
        int sum,done=0;
        starta=a;
        a=a->link;
        b=b->link;
        c=getnode();
        c->coef=c->x=c->y=c->z=-1;
        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){
        NODE cur;
        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;
        NODE cur;

        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;
        NODE a=NULL,b,c;
        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

```
→ #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();
}
```

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;
}
```