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

typedef struct NODE
{
   int data;
   struct NODE *next;
   struct NODE *prev;
}*node;

node getnode();
node insert_front(node head);
node insert_rear(node head);
void display_forward(node head);
void display_backward(node head);
node delete_front(node head);
node delete_rear(node head);
int count_nodes(node head);
void search(node head);
node insert_position(node head);
node delete_position(node head);
node search_and_insert_After(node head);
node search_and_insert_Before(node head);
node search_and_delete_After(node head);
node search_and_delete(node head);

main()
{
   node head = NULL;

   int choice,pos,c;
```

```
   for(;;)
   {
      printf("1-Insert Front\n2-Insert Rear\n3-Display_Foward\n4-Display_Backward\n5-
Delete_front\n6-Delete_rear\n7-Count_Nodes\n8-Search\n9-Insert_Position\n10-
Delete_Position\n11-Search_Insert_After\n12-Search_Insert_Before\n13-Search_Delete_After\n14-
Search_Delete\n15-Exit\n");
      scanf("%d",&choice);

      switch(choice)
      {
         case 1:head=insert_front(head);
         break;

         case 2:head=insert_rear(head);
         break;

         case 3:display_forward(head);
         break;

         case 4:display_backward(head);
         break;

         case 5:head=delete_front(head);
         break;

         case 6:head=delete_rear(head);
         break;

         case 7:c=count_nodes(head);
         printf("count=%d\n",c);
         break;
```

```
            case 8:search(head);
            break;

            case 9:head=insert_position(head);
            break;

            case 10:head=delete_position(head);
            break;

            case 11:head=search_and_insert_After(head);
            break;

            case 12:head=search_and_insert_Before(head);
            break;

            case 13:head=search_and_delete_After(head);
            break;

            case 14:head=search_and_delete(head);
            break;

           default:exit(0);

        }
     }

}

node getnode()
{
```

```
    node new_node;
    new_node=(node)malloc(sizeof(struct NODE));

    if(new_node==NULL)
    {
       printf("Not created\n");
       exit(0);
    }
    else
    {
       printf("Enter data to be Inserted\n");
       scanf("%d",&new_node->data);
       new_node->next = new_node;
       new_node->prev = new_node;
    }
    return new_node;
}

node insert_front(node head)
{
    node new_node;
    new_node=getnode();

    if(head==NULL)
    {
       head=new_node;
       return head;
    }
    else
    {
       head->prev->next=new_node;
```

```
        new_node->prev=head->prev;
        new_node->next=head;
        head->prev=new_node;
        head=new_node;
    }
   return head;
}

node insert_rear(node head)
{
    node new_node,cur;
    new_node=getnode();

    if(head==NULL)
    {
      head=new_node;
      return head;
    }
    else
    {
        cur=head->prev;
        cur->next=new_node;
        new_node->prev=cur;
        new_node->next=head;
        head->prev=new_node;

    }
    return head;
}

void display_forward(node head)
```

```
{
   node cur;

   if(head==NULL)
   {
      printf("CDLL is empty\n");
   }
   else
   {
      cur=head;
      while(cur->next!=head)
      {
      printf("Data=%d\n",cur->data);
      cur=cur->next;
      }
      printf("Data=%d\n",cur->data);
   }

}

void display_backward(node head)
{
   node cur;

   if(head==NULL)
   {
      printf("CDLL is empty\n");
   }
   else
   {
      cur=head;
```

```
        while(cur->next!=head)
        {
        cur=cur->next;
        }
        while(cur!=head)
        {
          printf("Data=%d\n",cur->data);
          cur=cur->prev;
        }
        printf("Data=%d\n",cur->data);

    }

}

node delete_front(node head)
{
    node cur;
    if(head==NULL)
    {
        printf("DLL is empty\n");
    }
    else if(head->next==head && head->prev==head)
    {
        printf("Data deletetd is\n");
        printf("Data=%d\n",head->data);
        free(head);
        head=NULL;
    }
    else
    {
```

```
        cur=head;
        head=head->next;
        head->prev=cur->prev;
        cur->prev->next=head;
        printf("Data deletetd is\n");
        printf("Data=%d\n",cur->data);
        free(cur);

    }
   return head;

}

node delete_rear(node head)
{
    node cur;
    if(head==NULL)
    {
        printf("DLL is empty\n");
    }
    else if(head->next==head && head->prev==head)
    {
        printf("Data deletetd is\n");
        printf("Data=%d\n",head->data);
        free(head);
        head=NULL;
    }
    else
    {
        cur=head->prev;
```

```
        printf("Data deletetd is\n");
        printf("Data=%d\n",cur->data);
        cur->prev->next=head;
        head->prev=cur->prev;
        free(cur);

    }
    return head;
}

int count_nodes(node head)
{
    node cur;
    int count=0;
    if(head==NULL)
        return(count);
    else
    {
        cur=head;
        while(cur->next!=head)
        {
            count++;
            cur=cur->next;
        }
        count++;
        return(count);
    }
}

void search(node head)
{
```

```
    node cur;
    int data;
    int found=0;
    printf("Enter data to be search\n");
    scanf("%d",&data);

    if(head==NULL)
       printf("CDLL is empty.....cant search\n");
    else
    {
        cur=head;
        while(cur->next!=head)
        {
           if(cur->data==data)
           {
              printf("Data found\n");
              printf("Data=%d\n",cur->data);
              found=1;
           }
           cur=cur->next;
        }
        if(cur->data==data)
           {
              printf("Data found\n");
              printf("Data=%d\n",cur->data);
              found=1;
           }
        if(found==0)
          printf("Data not found\n");
     }
}
```

```
node insert_position(node head)
{
   int pos,count;
   node new_node,cur;
   printf("Enter the position\n");
   scanf("%d",&pos);

   if(pos<=0 || pos>count_nodes(head)+1)
      printf("Invalid position\n");

   else if(pos==1)
      head=insert_front(head);

   else if(pos==count_nodes(head)+1)
      head=insert_rear(head);

   else
   {
      new_node=getnode();
      cur=head;
      count=1;
      while(count!=pos-1)
      {
         cur=cur->next;
         count++;
      }
      new_node->next=cur->next;
      cur->next->prev=new_node;
      cur->next=new_node;
      new_node->prev=cur;
```

```
    }
    return head;
}

node delete_position(node head)
{
    int pos,count;
    node cur,cur1;
    printf("Enter the position\n");
    scanf("%d",&pos);

    if(pos<=0 || pos>count_nodes(head))
        printf("Invalid position\n");

    else if(pos==1)
        head=delete_front(head);

    else if(pos==count_nodes(head))
        head=delete_rear(head);

    else
    {

        cur=head;
        count=1;
        while(count!=pos-1)
        {
            cur=cur->next;
            count++;
        }
        cur1=cur->next;
```

```
        printf("Data deleted is\n");
        printf("Data=%d\n",cur1->data);
        cur->next=cur1->next;
        cur1->next->prev=cur;
        free(cur1);
    }
    return head;
}



node search_and_insert_After(node head)
{
    int data, new_data;
    printf("Enter the data to be searched\n");
    scanf("%d", &data);

    if (head == NULL)
    {
        printf("List Empty\n");
        return head;
    }

    node cur = head;
    do
    {
        if (cur->data == data)
        {
            node new_node = getnode(new_data);
            new_node->prev = cur;
            new_node->next = cur->next;
```

```
            cur->next->prev = new_node;
            cur->next = new_node;

            return head;
        }
        cur = cur->next;
    } while (cur != head);

    printf("Data %d not found\n", data);
    return head;
}

node search_and_insert_Before(node head)
{
    int search_data, new_data;
    printf("Enter the data to be searched\n");
    scanf("%d", &search_data);

    if (head == NULL)
    {
        printf("List Empty\n");
        return head;
    }

    node cur = head;
    do
    {
        if (cur->data == search_data)
        {
            node new_node = getnode(new_data);
```

```
            new_node->next = cur;
            new_node->prev = cur->prev;

            cur->prev->next = new_node;
            cur->prev = new_node;

            if (cur == head)
               head = new_node;
            return head;
         }
         cur = cur->next;
      } while (cur != head);

      printf("Data %d not found\n", search_data);
      return head;
}

node search_and_delete_After(node head)
{
      int data;
      printf("Enter the data to be searched\n");
      scanf("%d", &data);

      if (head == NULL)
      {
         printf("List Empty\n");
         return head;
      }

      if (head->next == head)
      {
```

```
    if (head->data == data)
    {
        free(head);
        printf("Node with data %d deleted\n", data);
        return NULL;
    }
    else
    {
        printf("Data %d not found\n", data);
        return head;
    }
}

node cur = head;
do
{
    if (cur->data == data)
    {
        node del_node = cur->next;
        if (del_node == head)
        {
            node last_node = head->prev;
            head = del_node->next;
            last_node->next = head;
            head->prev = last_node;
            free(del_node);
            printf("Node with data %d deleted\n", data);
            return head;
        }
        else
        {
```

```
            cur->next = del_node->next;
            del_node->next->prev = cur;
            free(del_node);
            printf("Node after data %d deleted\n", data);
            return head;
          }
        }
        cur = cur->next;
    } while (cur != head);

    printf("Data %d not found\n", data);
    return head;
}


node search_and_delete(node head)
{
    int data;
    int found = 0;
    printf("Enter the data to be searched and deleted: ");
    scanf("%d", &data);

    if (head == NULL)
    {
        printf("List Empty\n");
        return head;
    }
    node cur = head;
    node temp;

    if (head->next == head)
```

```
{
    if (head->data == data)
    {
        printf("Deleted data: %d\n", head->data);
        free(head);
        return NULL;
    }
   else
    {
        printf("Data not found in the list\n");
        return head;
    }
}

while (cur->next != head)
{
    if (cur->data == data)
    {
        found = 1;
        temp = cur;



        if (cur == head)
        {
            head = head->next;
        }

        cur->prev->next = cur->next;
        cur->next->prev = cur->prev;
        printf("Deleted data: %d\n", cur->data);
        cur = cur->next;
```

```
        free(temp);
    }
    else
    {
        cur = cur->next;
    }
}

if (cur->data == data)
{
    found = 1;
    temp = cur;


    if (cur == head)
    {
        head = head->next;
    }

    cur->prev->next = cur->next;
    cur->next->prev = cur->prev;
    printf("Deleted data: %d\n", cur->data);
    free(temp);
}

if (found == 0)
{
    printf("%d data not in the list to delete\n", data);
}

return head;
```

}