Single Linked Lists And Circular Linked Lists

1. Create a single linked list and insert a node at the beginning, end and any position in between.

Source Code

```c
//
//  main.c
//  Lab-2-1
//
//  Created by Aaron R on 17/08/21.
//  Create a single linked list and insert a node at the beginning, end and any position in between.

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

struct node
{
    int data;
    struct node * link;
};

void display (struct node *temp)
{
    // Function to display the data in the linked list

    // If head is NULL, the linked list is empty
    if (temp==NULL)
        printf("The linked list is empty\n");
    else
    // Else, display it's elements
        while ((temp)!=NULL)
        {
            printf("%d\n",(temp)->data);
            temp = (temp)->link;
        }
}

void insert_head(struct node ** current, struct node ** head,int * count)
{
    // Function to insert node at the head
    *current = (struct node *) malloc(sizeof(struct node));

    // Input the node's data
    printf("Enter the data: ");
    scanf("%d",&(*current)->data);

    // The address part of the new node points to the previous head
    (*current)->link = *head;

    // The head variable is updated to the current one
    *head = *current;

    *count+=1;
}
```

```c
void insert_end(struct node ** current, struct node ** head,int * count)
{
    // Function to insert node at the end
    *current = (struct node *) malloc(sizeof(struct node));
    // Input the node's data
    printf("Enter the data: ");
    scanf("%d",&(*current)->data);

    // The link of the new node is NULL
    (*current)->link = NULL;

    // A temporary node is created
    struct node * temp;
    temp = *head;

    // It is used to traverse to the end of the existing linked list
    while (temp->link!=NULL)
    {
        temp = temp->link;
    }

    // The link of the node at the end of the existing linked list is updated to the new node
    temp->link = *current;

    *count+=1;
}

void insert_position(struct node ** current, struct node ** head,int * count,int p)
{
    // Function to insert node at the head
    *current = (struct node *) malloc(sizeof(struct node));

    // Input the node's data
    printf("Enter the data: ");
    scanf("%d",&(*current)->data);

    // A temporary node is created
    struct node * temp;

    // The temporary node is used to traverse and get to the pth position
    temp = *head;
    int i=1;
    while (i<p)
    {
        temp = temp->link;
        i+=1;
    }

    // The link of the new node is updated to the link of the node at (p+1)th position
    (*current)->link = temp->link;

    // The link of the node at the pth position is updated to the new node
    temp->link = (*current);

    *count+=1;
}
```

```c

int main(int argc, const char * argv[])
{
    // Initialising the linked list
    printf("Initialising a linked list...\n");
    struct node *head, *current;
    int count=0;
    head = NULL;
    current = NULL;

    // Displaying the initial elements of the linked list
    printf("Elements of the linked list:\n");
    display(head);

    int c=1;
    while (c==1)
    {
        int p;
        printf("\n~##### Insertion of data into linked list #####~\n");
        printf("1) 0 to insert a new node at the head\n2) %d or -1 to insert a new node at the end\nThere are currently %d nodes in the linked list\nEnter the position: ",count,count);
        scanf("%d",&p);

        if (p==0)
            insert_head(&current,&head,&count);

        else if (p==count||p==-1)
            insert_end(&current,&head,&count);

        else if((p>=0)&&(p<=count))
            insert_position(&current, &head, &count,p);

        else
            printf("Invalid position\n");

        printf("Elements of the linked list:\n");
        display(head);

        printf("To insert another node, press 1: ");
        scanf("%d",&c);

    }
    printf("\n---##############################---\n            ~End.\n\n");
    return 0;
}
```

Output(s):

```
Initialising a linked list...
Elements of the linked list:
The linked list is empty

~##### Insertion of data into linked list #####~
1) 0 to insert a new node at the head
2) 0 or -1 to insert a new node at the end
There are currently 0 nodes in the linked list
Enter the position: 0
Enter the data: 10
Elements of the linked list:
10
To insert another node, press 1: 1

~##### Insertion of data into linked list #####~
1) 0 to insert a new node at the head
2) 1 or -1 to insert a new node at the end
There are currently 1 nodes in the linked list
Enter the position: 1
Enter the data: 20
Elements of the linked list:
10
20
To insert another node, press 1: 1

~##### Insertion of data into linked list #####~
1) 0 to insert a new node at the head
2) 2 or -1 to insert a new node at the end
There are currently 2 nodes in the linked list
Enter the position: -1
Enter the data: 30
Elements of the linked list:
10
20
30
To insert another node, press 1: 1

~##### Insertion of data into linked list #####~
1) 0 to insert a new node at the head
2) 3 or -1 to insert a new node at the end
There are currently 3 nodes in the linked list
Enter the position: 2
Enter the data: 40
Elements of the linked list:
10
20
40
30
To insert another node, press 1: 1

~##### Insertion of data into linked list #####~
1) 0 to insert a new node at the head
2) 4 or -1 to insert a new node at the end
There are currently 4 nodes in the linked list
Enter the position: 5
Invalid position
Elements of the linked list:
10
20
40
30
To insert another node, press 1: 1
```

```
~##### Insertion of data into linked list #####~
1) 0 to insert a new node at the head
2) 4 or -1 to insert a new node at the end
There are currently 4 nodes in the linked list
Enter the position: 1
Enter the data: 50
Elements of the linked list:
10
50
20
40
30
To insert another node, press 1: 0

---##############################---
           ~End.

Program ended with exit code: 0
```

All Output Filter

2. Create a single linked list and delete a node at the beginning, end and any position in between.

Source Code

```c
//
//  main.c
//  Lab-2-2
//
//  Created by Aaron R on 18/08/21.
//  Create a single linked list and delete a node at the beginning, end and any position in between.

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

struct node
{
    int data;
    struct node * link;
};

void display (struct node *temp)
{
    // Function to display the data in the linked list

    // If head is NULL, the linked list is empty
    if (temp==NULL)
        printf("The single linked list is empty\n");
    else
    // Else, display it's elements
        while ((temp)!=NULL)
        {
            printf("%d\n",(temp)->data);
            temp = (temp)->link;
        }
}

// ######## Functions to insert nodes ##########

void insert_head(struct node ** current, struct node ** head,int * count)
{
    // ##### Function to insert node at the head #####
    *current = (struct node *) malloc(sizeof(struct node));

    // Input the node's data
    printf("Enter the data: ");
    scanf("%d",&(*current)->data);

    // The address part of the new node points to the previous head
    (*current)->link = *head;

    // The head variable is updated to the current one
    *head = *current;

    *count+=1;
}

```

```c
void insert_end(struct node ** current, struct node ** head,int * count)
{
    // ##### Function to insert node at the end #####

    if (*head==NULL)
    {
        insert_head(current,head,count);
    }

    else
    {
        *current = (struct node *) malloc(sizeof(struct node));
        // Input the node's data
        printf("Enter the data: ");
        scanf("%d",&(*current)->data);

        // The link of the new node is NULL
        (*current)->link = NULL;

        // A temporary node is created
        struct node * temp;
        temp = *head;

        // It is used to traverse to the end of the existing linked list
        while (temp->link!=NULL)
        {
            temp = temp->link;
        }

        // The link of the node at the end of the existing linked list is updated to the new node
        temp->link = *current;

        *count+=1;
    }
}

void insert_position(struct node ** current, struct node ** head,int * count,int p)
{
    // ##### Function to insert node at the head #####
    *current = (struct node *) malloc(sizeof(struct node));

    // Input the node's data
    printf("Enter the data: ");
    scanf("%d",&(*current)->data);

    // A temporary node is created
    struct node * temp;

    // The temporary node is used to traverse and get to the pth position
    temp = *head;
    int i=1;
    while (i<p)
    {
        temp = temp->link;
        i+=1;
    }

    // The link of the new node is updated to the link of the node at (p+1)th position
    (*current)->link = temp->link;

    // The link of the node at the pth position is updated to the new node
    temp->link = (*current);

    *count+=1;
}
```

```c
// ######## Functions to delete nodes ##########
void delete_head(struct node **head, int * count)
{
    // #### Function to delete the node at the beginning ####
    struct node * temp;
    temp=*head;
    // The address of the head is updated to the next node
    *head = (*head)->link;

    // free the memory of the previous tail
    free(temp);
    *count-=1;
}

void delete_end(struct node **head,int * count)
{
    // #####  If there is only one node, delete head #####
    if ((*head)->link==NULL)
        delete_head(head, count);

    // if there is more than one node
    else
    {
    struct node * temp;
    struct node * prev_temp;
    temp = *head;
    prev_temp=NULL;
    // Traverse through the single linked list and store the last node in temp
    // and the node preceeding it in prev_temp
    while (temp->link!=NULL)
    {
        prev_temp = temp;
        temp = temp->link;
    }

    // Update the link of the node preceeding the previous tail to NULL
    prev_temp->link = NULL;

    // Free the memory of the previous tail
    free(temp);
    *count-=1;
    }
}
void delete_pos(struct node **head, int p,int * count)
{
    // ##### Function to delete node at pth position #####

    struct node * temp;
    struct node * next_temp;
    temp = *head;

    // Traverse till the p-1th element and store it in temp
    int i=1;
    while (i<p-1)
    {
        temp = temp->link;
        i+=1;
    }
    // The pth node is stored in next_temp
    next_temp = temp->link;

    // The link of the p-1th node is updated to the p+1th node
    temp->link = next_temp->link;

    // free the memory of the previous tail
    free(next_temp);
    *count-=1;

}
```

```c

int main(int argc, const char * argv[])
{
    // Initialising the linked list
    printf("Initialising a single linked list...\n");
    struct node *head, *current;
    int count=0;
    head = NULL;
    current = NULL;

    // Displaying the initial elements of the linked list
    printf("Elements of the single linked list:\n");
    display(head);

    int c=1;
    while (c==1)
    {
        int p;
        printf("\n~##### Insertion of data into single linked list #####~\n");
        printf("\n1) 0 to insert a new node at the head\n2) %d or -1 to insert a new node at the end\nThere are currently %d nodes in the linked list\nEnter the position: ",count,count);
        scanf("%d",&p);

        if (p==0)
            insert_head(&current,&head,&count);

        else if (p==count||p==-1)
            insert_end(&current,&head,&count);

        else if((p>=0)&&(p<=count))
            insert_position(&current, &head, &count,p);

        else
            printf("Invalid position\n");

        printf("Elements of the linked list:\n");
        display(head);

        printf("To insert another node, press 1: ");
        scanf("%d",&c);

    }

    c=1;
    while (c==1)
    {
        if (head==NULL)
        {
            printf("The linked list is empty. There are no more nodes to delete.\n");
            break;
        }
        else
        {
            int p;
            printf("\n~##### Deletion of data in linked list #####~\n");
            printf("\n1) 0 to delete a new node at the head\n2) %d or -1 to delete a new node at the end\nThere are currently %d nodes in the linked list\nEnter the position in which to delete the new node: ",count,count);
            scanf("%d",&p);
            if (p==0)
                delete_head(&head,&count);

            else if ((p==count)||(p==-1))
                delete_end(&head,&count);

            else if((p>=0)&&(p<=count))
                delete_pos(&head, p,&count);

            else
                printf("Invalid position");
```

```
255
256             printf("Elements of the circular linked list:\n");
257             display(head);
258             printf("To delete another node, press 1: ");
259             scanf("%d",&c);
260
261         }
262
263     }
264
265     printf("\n---##############################---\n           ~End.\n\n");
266     return 0;
267 }
```

Outputs:

```
Initialising a single linked list...
Elements of the single linked list:
The single linked list is empty

~##### Insertion of data into single linked list #####~

1) 0 to insert a new node at the head
2) 0 or -1 to insert a new node at the end
There are currently 0 nodes in the linked list
Enter the position: -1
Enter the data: 10
Elements of the linked list:
10
To insert another node, press 1: 1

~##### Insertion of data into single linked list #####~

1) 0 to insert a new node at the head
2) 1 or -1 to insert a new node at the end
There are currently 1 nodes in the linked list
Enter the position: -1
Enter the data: 20
Elements of the linked list:
10
20
To insert another node, press 1: 1

~##### Insertion of data into single linked list #####~

1) 0 to insert a new node at the head
2) 2 or -1 to insert a new node at the end
There are currently 2 nodes in the linked list
Enter the position: -1
Enter the data: 30
Elements of the linked list:
10
20
30
To insert another node, press 1: 1
```

```
~##### Insertion of data into single linked list #####~

1) 0 to insert a new node at the head
2) 3 or -1 to insert a new node at the end
There are currently 3 nodes in the linked list
Enter the position: -1
Enter the data: 40
Elements of the linked list:
10
20
30
40
To insert another node, press 1: 1

~##### Insertion of data into single linked list #####~

1) 0 to insert a new node at the head
2) 4 or -1 to insert a new node at the end
There are currently 4 nodes in the linked list
Enter the position: -1
Enter the data: 50
Elements of the linked list:
10
20
30
40
50
To insert another node, press 1: 2

~##### Deletion of data in linked list #####~

1) 0 to delete a new node at the head
2) 5 or -1 to delete a new node at the end
There are currently 5 nodes in the linked list
Enter the position in which to delete the new node: 0
Elements of the circular linked list:
20
30
40
50
To delete another node, press 1: 1

~##### Deletion of data in linked list #####~

1) 0 to delete a new node at the head
2) 4 or -1 to delete a new node at the end
There are currently 4 nodes in the linked list
Enter the position in which to delete the new node: 2
Elements of the circular linked list:
20
40
50
To delete another node, press 1: 1
```

```
~##### Deletion of data in linked list #####~

1) 0 to delete a new node at the head
2) 3 or -1 to delete a new node at the end
There are currently 3 nodes in the linked list
Enter the position in which to delete the new node: -1
Elements of the circular linked list:
20
40
To delete another node, press 1: 1

~##### Deletion of data in linked list #####~

1) 0 to delete a new node at the head
2) 2 or -1 to delete a new node at the end
There are currently 2 nodes in the linked list
Enter the position in which to delete the new node: 2
Elements of the circular linked list:
20
To delete another node, press 1: 1

~##### Deletion of data in linked list #####~

1) 0 to delete a new node at the head
2) 1 or -1 to delete a new node at the end
There are currently 1 nodes in the linked list
Enter the position in which to delete the new node: 5
Invalid position
Elements of the circular linked list:
20
To delete another node, press 1: 1

~##### Deletion of data in linked list #####~

1) 0 to delete a new node at the head
2) 1 or -1 to delete a new node at the end
There are currently 1 nodes in the linked list
Enter the position in which to delete the new node: 0
Elements of the circular linked list:
The single linked list is empty
To delete another node, press 1: 1
The linked list is empty. There are no more nodes to delete.

---##############################---
            ~End.

Program ended with exit code: 0
```

All Output ⌃ Filter

3. Create a circular single linked list and insert a node at the beginning, end and any position in between.

Source code

```c
//
//  main.c
//  Lab-2-3
//
//  Created by Aaron R on 18/08/21.
//  Create a circular single linked list and insert a node at the beginning, end and any position in between

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

struct node
{
    int data;
    struct node * link;
}*tail;

void display(void)
{
    // #### To display the elements of the circular linked list ####
    struct node * temp;
    temp = tail->link;
    if (tail==NULL)
    {
        printf("The circular linked list is empty");
    }
    else
    {
        // traversing from the first node
        while (temp->link!=tail->link)
        {
            printf("%d\n",temp->data);
            temp = temp->link;
        }
        // printing the last node's data
        printf("%d\n",temp->data);
    }
}
void insert_head(int * count)
{
    // #### Function to insert node at the head ####
    struct node * current;
    current = (struct node *) malloc(sizeof(struct node));
    // Inputting data to the node
    printf("Enter data: ");
    scanf("%d",&current->data);
    current->link=NULL;

    // If the circular linked list is empty, assign the new node as the tail
    if (tail == NULL)
    {
        tail = current;
        tail->link = current;
    }
    // If the circular linked list isn't empty
    else
    {
        // The new node is linked to the head
        current->link = tail->link;

        // The tail is linked to the new node
        tail->link = current;
    }

    *count +=1;
}
```

```c
void insert_end(int * count)
{
    // #### Function to insert node at the end ####
    struct node * current;
    current = (struct node *) malloc(sizeof(struct node));
    // Inputting data to the node
    printf("Enter the data: ");
    scanf("%d",&current->data);
    current->link = NULL;

    // If the circular linked list is empty, assign the new node as the tail
    if (tail==NULL)
    {
        tail = current;
        tail->link = current;
    }

    // If the circular linked list isn't empty
    else
    {
        // The new node's link is assigned the address of the head
        current->link = tail->link;

        // The previous tail's link is assigned the address of the new node
        tail->link = current;

        // The tail variable is updated to the new node
        tail = current;
    }
    *count+=1;

}

void insert_pos(int * count, int p)
{
    // #### Function to insert node in position p ####
    struct node * current;
    struct node * temp;
    current = (struct node *) malloc(sizeof(struct node));

    // Inputting data to the node
    printf("Enter data: ");
    scanf("%d",&current->data);
    current->link=NULL;

    // The temp variable is assigned the address of the head
    temp = tail->link;
    int i=0;

    // The temp variable traverses untill it reaches the p-1th position
    while (i<p-1)
    {
        temp = temp->link;
        i+=1;
    }
    // The new node's link is given the address of the p+1th element
    current->link = temp->link;

    // The p-1th element's link is given to be the address of the new node
    temp->link = current;

    *count+=1;
}
```

```c
int main(int argc, const char * argv[])
{

    int count=0;
    int c=1;
    while (c==1)
    {
        int p;
        printf("\n~##### Insertion of data into circular linked list #####~\n");
        printf("\n1) 0 to insert a new node at the head\n2) %d or -1 to insert a new node at the end\nThere are
            currently %d nodes in the linked list\nEnter the position in which to insert the new node:
            ",count,count);
        scanf("%d",&p);
        if (p==0)
        {
            insert_head(&count);
        }
        else if (p==count||p==-1)
        {
            insert_end(&count);
        }
        else if((p>=0)&&(p<=count))
        {
            insert_pos(&count,p);
        }
        else
            printf("Invalid position\n");

        printf("Elements of the circular linked list:\n");
        display();

        printf("To insert another node, press 1: ");
        scanf("%d",&c);
    }
    printf("\n---##############################---\n           ~End.\n\n");
    return 0;
}
```

Outputs:

```
~##### Insertion of data into circular linked list #####~

1) 0 to insert a new node at the head
2) 0 or -1 to insert a new node at the end
There are currently 0 nodes in the linked list
Enter the position in which to insert the new node: 0
Enter data: 10
Elements of the circular linked list:
10
To insert another node, press 1: 1

~##### Insertion of data into circular linked list #####~

1) 0 to insert a new node at the head
2) 1 or -1 to insert a new node at the end
There are currently 1 nodes in the linked list
Enter the position in which to insert the new node: 1
Enter the data: 20
Elements of the circular linked list:
10
20
To insert another node, press 1: 1

~##### Insertion of data into circular linked list #####~

1) 0 to insert a new node at the head
2) 2 or -1 to insert a new node at the end
There are currently 2 nodes in the linked list
Enter the position in which to insert the new node: -1
Enter the data: 30
Elements of the circular linked list:
10
20
30
To insert another node, press 1: 1

~##### Insertion of data into circular linked list #####~

1) 0 to insert a new node at the head
2) 3 or -1 to insert a new node at the end
There are currently 3 nodes in the linked list
Enter the position in which to insert the new node: 2
Enter data: 40
Elements of the circular linked list:
10
20
40
30
To insert another node, press 1: 1

~##### Insertion of data into circular linked list #####~

1) 0 to insert a new node at the head
2) 4 or -1 to insert a new node at the end
There are currently 4 nodes in the linked list
Enter the position in which to insert the new node: 6
Invalid position
Elements of the circular linked list:
10
20
40
30
To insert another node, press 1: 0

---##############################---
           ~End.

Program ended with exit code: 0
```

All Output ⌃ Filter

4. Create a circular single linked list and delete a node at the beginning, end and any position in between.

Source Code

```c
//
//  main.c
//  Lab-2-4
//
//  Created by Aaron R on 19/08/21.
//  Create a circular single linked list
//  and delete a node at the beginning, end and any position in between.

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

struct node
{
    int data;
    struct node * link;
}*tail;

void display(void)
{
    // #### To display the elements of the circular linked list ####
    struct node * temp;

    if (tail==NULL)
    {
        printf("The circular linked list is empty\n");
    }
    else
    {
        temp = tail->link;
        // traversing from the first node
        while (temp->link!=tail->link)
        {
            printf("%d\n",temp->data);
            temp = temp->link;
        }
        // printing the last node's data
        printf("%d\n",temp->data);
    }
}

// ########## Functions to insert nodes ##########
void insert_head(int * count)
{
    // #### Function to insert node at the head ####
    struct node * current;
    current = (struct node *) malloc(sizeof(struct node));
    // Inputting data to the node
    printf("Enter data: ");
    scanf("%d",&current->data);
    current->link=NULL;

    // If the circular linked list is empty, assign the new node as the tail
    if (tail == NULL)
    {
        tail = current;
        tail->link = current;
    }
    // If the circular linked list isn't empty
    else
    {
        // The new node is linked to the head
        current->link = tail->link;

        // The tail is linked to the new node
        tail->link = current;
    }

    *count +=1;
}
```

```c

void insert_end(int * count)
{
    // #### Function to insert node at the end ####
    struct node * current;
    current = (struct node *) malloc(sizeof(struct node));
    // Inputting data to the node
    printf("Enter the data: ");
    scanf("%d",&current->data);
    current->link = NULL;

    // If the circular linked list is empty, assign the new node as the tail
    if (tail==NULL)
    {
        tail = current;
        tail->link = current;
    }

    // If the circular linked list isn't empty
    else
    {
        // The new node's link is assigned the address of the head
        current->link = tail->link;

        // The previous tail's link is assigned the address of the new node
        tail->link = current;

        // The tail variable is updated to the new node
        tail = current;
    }
    *count+=1;

}

void insert_pos(int * count, int p)
{
    // #### Function to insert node in position p ####
    struct node * current;
    struct node * temp;
    current = (struct node *) malloc(sizeof(struct node));

    // Inputting data to the node
    printf("Enter data: ");
    scanf("%d",&current->data);
    current->link=NULL;

    // The temp variable is assigned the address of the head
    temp = tail->link;
    int i=1;

    // The temp variable traverses untill it reaches the p-1th position
    while (i<p-1)
    {
        temp = temp->link;
        i+=1;
    }
    // The new node's link is given the address of the p+1th element
    current->link = temp->link;

    // The p-1th element's link is given to be the address of the new node
    temp->link = current;

    *count+=1;
}
```

```c

// ########## Functions to delete nodes ##########
void delete_head(int * count)
{
    // #### Function to delete a node at the beginning ####

    struct node * temp;
    // Initialise the temp variable with the head of the circular linked list
    temp = tail->link;

    // If the tail is empty, display that the circular linked list is empty
    if (tail==0)
        printf("The circular linked list is empty");

    // If the tail has only one node, delete the node
    else if (tail->link==tail)
        tail = NULL;


    // If the tail has more than one node,
    // update the link of the tail node to the 2nd node of circular linked list
    else
    {
        tail->link = temp->link;
    }

    *count-=1;

}

void delete_end(int * count)
{
    // #### Function to delete a node at the end ####

    struct node * current;
    struct node * prev;
    prev=NULL;
    // Initialise the current variable with the head of the circular linked list
    current = tail->link;

    // If the tail is empty, display that the circular linked list is empty
    if (tail==NULL)
        printf("The circular linked list is empty\n");

    // If the tail has only one node, delete the node
    else if (tail->link==tail)
        tail = NULL;

    // If the tail has more than one node,
    else
    {
        // Traverse the list using the current variable till the tail is reached
        while (current->link!=tail->link)
        {
            // The element that preceeds the tail is stored in the variable prev
            prev = current;
            current = current->link;
        }
        // The link of the node preceeding the tail is updated to the address of the head
        prev->link = tail->link;

        // The tail variable is updated with the element that preceeded the previous tail
        tail = prev;
    }
    *count-=1;
    // free the memory of the node that was deleted
    free(current);
}
```

```c

void delete_pos(int * count, int p)
{
    // #### Function to delete a node at the pth position ####

    struct node * current;
    current = tail->link;
    struct node * next_current;
    next_current = NULL;
    int i=1;

    // Traverse till the p-1th element is reached and stored in the current variable
    while (i<p-1)
    {
        current = current->link;
        i++;
    }
    // The pth element is stored in the next_current variable
    next_current = current->link;

    // The link of the p-1th element is updated to the address of the p+1th element
    current->link = next_current->link;

    // free the memory of the node that was deleted
    free(current);
    *count-=1;
}

int main(int argc, const char * argv[])
{

    int count=0;

    // ########## Menu to insert nodes ##########
    int c=1;
    while (c==1)
    {
        int p;
        printf("\n~##### Insertion of data into linked list #####~\n");
        printf("Enter the position from which to delete the node\n1) 0 to insert a new node at the head\n2) %d or
            -1 to insert a new node at the end\nThere are currently %d nodes in the linked list\nEnter the
            position to insert: ",count,count);
        scanf("%d",&p);
        if (p==0)
        {
            insert_head(&count);
        }
        else if (p==count||p==-1)
        {
            insert_end(&count);
        }
        else if((p>0)&&(p<count))
        {
            insert_pos(&count,p);
        }

        printf("Elements of the circular linked list:\n");
        display();

        printf("To insert another node, press 1: ");
        scanf("%d",&c);
    }

    // ########## Menu to delete nodes ##########
    c=1;
    while (c==1)
    {
        if (tail==NULL)
        {
            printf("There are no more nodes left to delete.\n\n");
            break;
        }
```

```
272
273         else
274         {
275             int p;
276             printf("\n~##### Deletion of data from circular linked list #####~\n");
277             printf("Enter the position in which to insert the new node\n1) 0 to delete a new node at the head\n2)
                    %d or -1 to delete a new node at the end\nThere are currently %d nodes in the linked list\nEnter
                    the position to delete: ",count,count);
278             scanf("%d",&p);
279             if (p==0)
280             {
281                 delete_head(&count);
282             }
283             else if (p==count||p==-1)
284             {
285                 delete_end(&count);
286             }
287             else if((p>0)&&(p<count))
288             {
289                 delete_pos(&count, p);
290             }
291
292             else
293                 printf("Invalid position\n");
294
295             printf("Elements of the circular linked list:\n");
296             display();
297
298             printf("To delete another node, press 1: ");
299             scanf("%d",&c);
300         }
301     }
302     return 0;
303 }
```

Outputs

```
~##### Insertion of data into linked list #####~
Enter the position from which to delete the node
1) 0 to insert a new node at the head
2) 0 or -1 to insert a new node at the end
There are currently 0 nodes in the linked list
Enter the position to insert: 0
Enter data: 10
Elements of the circular linked list:
10
To insert another node, press 1: 1

~##### Insertion of data into linked list #####~
Enter the position from which to delete the node
1) 0 to insert a new node at the head
2) 1 or -1 to insert a new node at the end
There are currently 1 nodes in the linked list
Enter the position to insert: 0
Enter data: 20
Elements of the circular linked list:
20
10
To insert another node, press 1: 1

~##### Insertion of data into linked list #####~
Enter the position from which to delete the node
1) 0 to insert a new node at the head
2) 2 or -1 to insert a new node at the end
There are currently 2 nodes in the linked list
Enter the position to insert: 0
Enter data: 30
Elements of the circular linked list:
30
20
10
To insert another node, press 1: 1

~##### Insertion of data into linked list #####~
Enter the position from which to delete the node
1) 0 to insert a new node at the head
2) 3 or -1 to insert a new node at the end
There are currently 3 nodes in the linked list
Enter the position to insert: 0
Enter data: 40
Elements of the circular linked list:
40
30
20
10
To insert another node, press 1: 0
```

```
~##### Deletion of data from circular linked list #####~
Enter the position in which to insert the new node
1) 0 to delete a new node at the head
2) 4 or -1 to delete a new node at the end
There are currently 4 nodes in the linked list
Enter the position to delete: 2
Elements of the circular linked list:
40
20
10
To delete another node, press 1: 1

~##### Deletion of data from circular linked list #####~
Enter the position in which to insert the new node
1) 0 to delete a new node at the head
2) 3 or -1 to delete a new node at the end
There are currently 3 nodes in the linked list
Enter the position to delete: -1
Elements of the circular linked list:
40
20
To delete another node, press 1: 1

~##### Deletion of data from circular linked list #####~
Enter the position in which to insert the new node
1) 0 to delete a new node at the head
2) 2 or -1 to delete a new node at the end
There are currently 2 nodes in the linked list
Enter the position to delete: 0
Elements of the circular linked list:
20
To delete another node, press 1: 1

~##### Deletion of data from circular linked list #####~
Enter the position in which to insert the new node
1) 0 to delete a new node at the head
2) 1 or -1 to delete a new node at the end
There are currently 1 nodes in the linked list
Enter the position to delete: 1
Elements of the circular linked list:
The circular linked list is empty
To delete another node, press 1: 1
There are no more nodes left to delete.

Program ended with exit code: 0
```

All Output ⌃ Filter