// C Program to illustrate how to create a circular linked

// list

#include <stdio.h>

#include <stdlib.h>

// Define the structure for a node

struct Node {

int data;

struct Node\* next;

};

// Function to insert a new node at the end of the list

struct Node\* insertAtEnd(struct Node\* last, int data)

{

struct Node\* newNode

= (struct Node\*)malloc(sizeof(struct Node));

newNode->data = data;

if (last == NULL) {

// If the list is empty, make the new node the only

// node in the list

last = newNode;

newNode->next = last;

}

else {

// Add the new node to the end and update the last

// node's next pointer

newNode->next = last->next;

last->next = newNode;

last = newNode; // Update the last pointer to the

// new node

}

return last; // Return the updated last pointer

}

// Function to display the circular linked list

void display(struct Node\* last)

{

if (last == NULL) {

printf("List is empty.\n");

return;

}

struct Node\* temp

= last->next; // Start from the first node

do {

printf("%d ", temp->data);

temp = temp->next;

} while (temp != last->next);

printf("\n");

}

int main()

{

struct Node\* last

= NULL; // Initialize an empty circular linked list

// Insert nodes at the end

last = insertAtEnd(last, 10);

last = insertAtEnd(last, 20);

last = insertAtEnd(last, 30);

// Display the circular linked list

printf("Circular Linked List: ");

display(last);

return 0;

}

Output:

Circular Linked List: 10 20 30

//Menu-driven program in C implementing all operations on circular singly linked list

#include<stdio.h>

#include<stdlib.h>

struct node

{

**int** data;

    struct node \*next;

};

struct node \*head;

**void** beginsert ();

**void** lastinsert ();

**void** randominsert();

**void** begin\_delete();

**void** last\_delete();

**void** random\_delete();

**void** display();

**void** search();

**void** main ()

{

**int** choice =0;

**while**(choice != 7)

    {

        printf("\n\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\*\n");

        printf("\nChoose one option from the following list ...\n");

        printf("\n===============================================\n");

        printf("\n1.Insert in begining\n2.Insert at last\n3.Delete from Beginning\n4.Delete from last\n5.Search for an element\n6.Show\n7.Exit\n");

        printf("\nEnter your choice?\n");

        scanf("\n%d",&choice);

**switch**(choice)

        {

**case** 1:

            beginsert();

**break**;

**case** 2:

            lastinsert();

**break**;

**case** 3:

            begin\_delete();

**break**;

**case** 4:

            last\_delete();

**break**;

**case** 5:

            search();

**break**;

**case** 6:

            display();

**break**;

**case** 7:

            exit(0);

**break**;

**default**:

            printf("Please enter valid choice..");

        }

    }

}

**void** beginsert()

{

    struct node \*ptr,\*temp;

**int** item;

    ptr = (struct node \*)malloc(sizeof(struct node));

**if**(ptr == NULL)

    {

        printf("\nOVERFLOW");

    }

**else**

    {

        printf("\nEnter the node data?");

        scanf("%d",&item);

        ptr -> data = item;

**if**(head == NULL)

        {

            head = ptr;

            ptr -> next = head;

        }

**else**

        {

            temp = head;

**while**(temp->next != head)

                temp = temp->next;

            ptr->next = head;

            temp -> next = ptr;

            head = ptr;

        }

        printf("\nnode inserted\n");

    }

}

**void** lastinsert()

{

    struct node \*ptr,\*temp;

**int** item;

    ptr = (struct node \*)malloc(sizeof(struct node));

**if**(ptr == NULL)

    {

        printf("\nOVERFLOW\n");

    }

**else**

    {

        printf("\nEnter Data?");

        scanf("%d",&item);

        ptr->data = item;

**if**(head == NULL)

        {

            head = ptr;

            ptr -> next = head;

        }

**else**

        {

            temp = head;

**while**(temp -> next != head)

            {

                temp = temp -> next;

            }

            temp -> next = ptr;

            ptr -> next = head;

        }

        printf("\nnode inserted\n");

    }

}

**void** begin\_delete()

{

    struct node \*ptr;

**if**(head == NULL)

    {

        printf("\nUNDERFLOW");

    }

**else** **if**(head->next == head)

    {

        head = NULL;

        free(head);

        printf("\nnode deleted\n");

    }

**else**

    {   ptr = head;

**while**(ptr -> next != head)

            ptr = ptr -> next;

        ptr->next = head->next;

        free(head);

        head = ptr->next;

        printf("\nnode deleted\n");

    }

}

**void** last\_delete()

{

    struct node \*ptr, \*preptr;

**if**(head==NULL)

    {

        printf("\nUNDERFLOW");

    }

**else** **if** (head ->next == head)

    {

        head = NULL;

        free(head);

        printf("\nnode deleted\n");

    }

**else**

    {

        ptr = head;

**while**(ptr ->next != head)

        {

            preptr=ptr;

            ptr = ptr->next;

        }

        preptr->next = ptr -> next;

        free(ptr);

        printf("\nnode deleted\n");

    }

}

**void** search()

{

    struct node \*ptr;

**int** item,i=0,flag=1;

    ptr = head;

**if**(ptr == NULL)

    {

        printf("\nEmpty List\n");

    }

**else**

    {

        printf("\nEnter item which you want to search?\n");

        scanf("%d",&item);

**if**(head ->data == item)

        {

        printf("item found at location %d",i+1);

        flag=0;

        }

**else**

        {

**while** (ptr->next != head)

        {

**if**(ptr->data == item)

            {

                printf("item found at location %d ",i+1);

                flag=0;

**break**;

            }

**else**

            {

                flag=1;

            }

            i++;

            ptr = ptr -> next;

        }

        }

**if**(flag != 0)

        {

            printf("Item not found\n");

        }

    }

}

**void** display()

{

    struct node \*ptr;

    ptr=head;

**if**(head == NULL)

    {

        printf("\nnothing to print");

    }

**else**

    {

        printf("\n printing values ... \n");

**while**(ptr -> next != head)

        {

            printf("%d\n", ptr -> data);

            ptr = ptr -> next;

        }

        printf("%d\n", ptr -> data);

    }

}

**Output:**

*\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\**

*Choose one option from the following list ...*

*===============================================*

*1.Insert in begining*

*2.Insert at last*

*3.Delete from Beginning*

*4.Delete from last*

*5.Search for an element*

*6.Show*

*7.Exit*

*Enter your choice?*

*1*

*Enter the node data?10*

*node inserted*

*\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\**

*Choose one option from the following list ...*

*===============================================*

*1.Insert in begining*

*2.Insert at last*

*3.Delete from Beginning*

*4.Delete from last*

*5.Search for an element*

*6.Show*

*7.Exit*

*Enter your choice?*

*2*

*Enter Data?20*

*node inserted*

*\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\**

*Choose one option from the following list ...*

*===============================================*

*1.Insert in begining*

*2.Insert at last*

*3.Delete from Beginning*

*4.Delete from last*

*5.Search for an element*

*6.Show*

*7.Exit*

*Enter your choice?*

*2*

*Enter Data?30*

*node inserted*

*\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\**

*Choose one option from the following list ...*

*===============================================*

*1.Insert in begining*

*2.Insert at last*

*3.Delete from Beginning*

*4.Delete from last*

*5.Search for an element*

*6.Show*

*7.Exit*

*Enter your choice?*

*3*

*node deleted*

*\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\**

*Choose one option from the following list ...*

*===============================================*

*1.Insert in begining*

*2.Insert at last*

*3.Delete from Beginning*

*4.Delete from last*

*5.Search for an element*

*6.Show*

*7.Exit*

*Enter your choice?*

*4*

*node deleted*

*\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\**

*Choose one option from the following list ...*

*===============================================*

*1.Insert in begining*

*2.Insert at last*

*3.Delete from Beginning*

*4.Delete from last*

*5.Search for an element*

*6.Show*

*7.Exit*

*Enter your choice?*

*5*

*Enter item which you want to search?*

*20*

*item found at location 1*

*\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\**

*Choose one option from the following list ...*

*===============================================*

*1.Insert in begining*

*2.Insert at last*

*3.Delete from Beginning*

*4.Delete from last*

*5.Search for an element*

*6.Show*

*7.Exit*

*Enter your choice?*

*6*

*printing values ...*

*20*

*\*\*\*\*\*\*\*\*\*Main Menu\*\*\*\*\*\*\*\*\**

*Choose one option from the following list ...*

*===============================================*

*1.Insert in begining*

*2.Insert at last*

*3.Delete from Beginning*

*4.Delete from last*

*5.Search for an element*

*6.Show*

*7.Exit*

*Enter your choice?*

*7*