AIM: Implement Circular Linked List ADT.

CODE:

#include<stdio.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*next;

};

struct node \*head;

void begInsert();

void endInsert();

void begDelete();

void endDelete();

void count();

void display();

void main ()

{

int choice = 0;

clrscr();

while(choice != 7)

{

printf("\n\n---Implementation of Circular Linked List---\n\n");

printf("\n1.Insertion at beginning\n2.Insertion at end\n3.Deletion from beginning\n4.Deletion from end\n5.Number of nodes\n6.Display\n7.Exit\n");

printf("\nEnter your choice: ");

scanf("%d",&choice);

switch(choice)

{

case 1:

begInsert();

break;

case 2:

endInsert();

break;

case 3:

begDelete();

break;

case 4:

endDelete();

break;

case 5:

count();

break;

case 6:

display();

break;

case 7:

exit(0);

break;

default:

printf("Please enter valid choice");

}

}

getch();

}

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 endInsert()

{

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 begDelete()

{

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 endDelete()

{

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 count()

{

struct node \*ptr;

int count = 0;

ptr = head;

if(head == NULL)

{

printf("\nList is empty");

}

while(ptr != NULL)

{

count++;

ptr = ptr -> next;

}

printf("\nThe total number of nodes are: %d", count);

}

void display()

{

struct node \*ptr;

ptr=head;

if(head == NULL)

{

printf("\nNothing to print");

}

else

{

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

while(ptr -> next != head)

{

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

ptr = ptr -> next;

}

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

}

}

OUTPUT:











