**Name: Rohit Ghosh.**

**Roll Number: 21051420.**

**Date: 08/08/2022.**

**Class Assignment**

**Program 6**

**Question:**

Single Circular Linked List:

WAP to perform the following operations on a Single Linked List:

1. Create new Node
2. Insert a new Node
3. Delete a Node Both by Data and Position
4. Search for a data
5. Display

**Code:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <math.h>

#include <stdint.h>

#include <stdbool.h>

//all packages are imported

typedef struct node

{

int data;

struct node \*next;

}\*NODE;

//we made the structure

NODE create(NODE start)

{

//this method creates a new node

NODE new\_node;

int data;

printf("\nEnter the data. If entered -1, list is complete \n");

scanf("%d",&data);

NODE last=NULL;

while(data!=-1)

{

new\_node=(struct node\*)malloc(sizeof(struct node\*));

new\_node->data=data;

if(start==NULL) //making it the first node

{

//this is for empty list

start=new\_node;

new\_node->next=start;

last=new\_node;

}

else

{

//here the nodes are inserted from the first

last->next=new\_node;

last=new\_node;

new\_node->next=start;

}

printf("\nEnter the data. If entered -1, list is complete \n");

scanf("%d",&data);

}

return start;

}

//end of creating node method

void display(NODE start)

{

//this method displays the list

NODE ptr=start;

int count=1;

if(start==NULL)

{

printf("\nLIST IS EMPTY\n");

//for empty list

}

else

{

printf("\nThe List:\n");

do

{

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

ptr=ptr->next;

count++;

}while(ptr!=start);

//list is displayed

}

printf("\n");

}

//end of diaplay method

int count\_node(NODE start)

{

//here the number of nodes are counted

NODE ptr=start;

int count=0;

do

{

ptr=ptr->next;

count++;

}while(ptr!=start);

return count;

}

//end of count\_node method

NODE insert(NODE start)

{

//this method inserts an item entered by the user

if(start==NULL)

{

start=create(start);

display(start);

return start;

}//calling create as list is empty

int data;

printf("\nEnter the Data\n");

scanf("%d",&data);

NODE new\_node;

new\_node=(struct node\*)malloc(sizeof(struct node\*));

new\_node->data=data;

if(new\_node==NULL)

{

printf("\nMemory Overflow\n");

exit(0);

}

//overflow condition checked

label:

//label created

int pos;

printf("\nEnter the Position\n");

scanf("%d",&pos);

if(pos==1)

{

NODE ptr=start;

while(ptr->next!=start)

{

ptr=ptr->next;

}

new\_node->next=start;

ptr->next=new\_node;

start=new\_node;

start=new\_node;

//this condition is for first position or empty list

}

else

{

int count = count\_node(start);

if(pos==count+1)

{

//this is for last position

NODE ptr=start;

do

{

ptr=ptr->next;

}while(ptr->next!=start);

ptr->next=new\_node;

new\_node->next=start;

}

else if(pos>count+1||pos<1)

{

//this is for invalid position

printf("\nInvalid Position!! Try again\n");

goto label;

}

else

{

//this is for a valid position between the first and last position

NODE ptr=start;

int i=0;

while(i!=pos-2)

{

ptr=ptr->next;

i++;

}

new\_node->next=ptr->next;

ptr->next=new\_node;

}

}

display(start);

return start;

}

//end of insert method

NODE deleteByData(NODE start)

{

//this method deletes an item entered by the user

printf("\nEnter the Data to be deleted\n");

int data;

scanf("%d",&data);

if(start==NULL)

{

printf("\nUnderflow of List\n");

exit(0);

}

//underflow condition is checked

NODE ptr=start;

NODE ptr2=NULL;

int flag=0;

if(ptr->next==start)

{

start=NULL;

flag=1;

}//for only one node

else

{

while(ptr->next!=start)

{

if(ptr->data==data)

{

//first node

if(ptr==start)

{

NODE ptrtemp=start;

while(ptrtemp->next!=start)

{

ptrtemp=ptrtemp->next;

}

start=start->next;

ptrtemp->next=start;

}

//logic for interim delete

else

{

ptr2->next=ptr->next;

}

flag=1;

break;

}

else

{

ptr2=ptr;

ptr=ptr->next;

}

// logic for last node deletion

if(ptr->data==data)

{

ptr2->next=start;

flag=1;

}

}

}

if(flag==1)

{

display(start);

}

else

{

printf("\nItem Not Found!! Try Again!!\n");

}

return start;

}

//end of delete method

NODE deleteByPosition(NODE start)

{

//this method deletes data by accepting the position

label:

//label created

printf("\nEnter the Position to be Deleted\n");

int pos;

scanf("%d",&pos);

if(start==NULL)

{

printf("\nMemory Underflow\n");

exit(0);

}

//overflow condition checked

if(pos==1)

{

NODE ptr=start;

if(ptr->next==start)

{

start=NULL;

}

else

{

while(ptr->next!=start)

{

ptr=ptr->next;

}

start=start->next;

ptr->next=start;

}

//this condition is for deleting the first node

}

else

{

int count = count\_node(start);

if(pos>count+1||pos<1)

{

//this is for invalid position

printf("\nInvalid Position!! Try again\n");

goto label;

}

else

{

int copy=1;

NODE ptr=start;

NODE ptr2=NULL;

while(copy<=pos-1)

{

//printf("%d %d",copy,pos);

ptr2=ptr;

ptr=ptr->next;

copy++;

}

ptr2->next=ptr->next;

}

}

display(start);

return start;

}

//end of deleteByPosition Method

void search(NODE start,int data)

{

//this method searches an item entered by the user

NODE ptr=start;

int flag=0;//flag variable to check the variable found or not

do

{

if(ptr->data==data)

{

flag=1;

break;

//flag is updated to 1 for getting the item in the list

}

else

{

ptr=ptr->next;

}

}while(ptr!=start);

if(flag==0)

{

printf("\nItem Not Found\n");

}

else

{

printf("\nItem Found\n");

}

//appropriate message is shown

}

//end of search method

NODE delete\_check(NODE start)

{

//this method checks which type of delete the user wants

label:

//label created

printf("\n");

printf("\n1. Delete By Entering Data\n");

printf("2. Delete By Entering Position\n");

printf("\nEnter Your Choice:\n");

int response;

scanf("%d",&response);

switch (response)

{

case 1:

start=deleteByData(start);

break;

case 2:

start=deleteByPosition(start);

break;

default:

printf("\nWrong Choice!! Try Again!!\n");

goto label;

break;

}//end of switch case

return start;

}

//end of delete\_check method

int main()

{

//this is the main execution method which controls the flow of execution

NODE start=NULL;

//main execution function

for(;;)//this is an infinite for loop

{

int data;

printf("\n1. Create\n");

printf("2. Insert\n");

printf("3. Delete\n");

printf("4. Search\n");

printf("5. Display\n");

printf("6. Exit\n");

printf("\nEnter Your Choice\n\n");

//here the menu is displayed

int response;

scanf("%d",&response);

switch (response)

{

case 1:

start=create(start);

break;

case 2:

start=insert(start);

break;

case 3:

start=delete\_check(start);

break;

case 4:

printf("\nEnter the Data to be searched\n");

scanf("%d",&data);

search(start,data);

break;

case 5:

display(start);

break;

case 6:

exit(0);

break;

default:

printf("\nWRONG INPUT!! TRY AGAIN\n");

break;

}

//this is the switch case

}

return 0;

}

//end of main method

**Output:**











