Ex -no : 2
Date: 16-11-22

Name: M.Rohith
3122 21 5001 085

SSN College of Engineering

Department of Computer Science and Engineering

UCS2312 – Data Structures Lab

II Year CSE - B Section (III Semester)

Academic Year 2022-23

Staff Incharge : Dr.H.Shahul Hamead

Exercise 2: Exercises on Circular Linked List

Aim:

To implement basic C programs by using Circular Linked List.

Basic

Implement a circular linked list abstract data type (ADT) with Create, print, search, insert (at the middle, head), delete (at the middle, head) functions.

Ex -no : 2
Date: 16-11-22

Name: M.Rohith
3122 21 5001 085

Pseudocode:

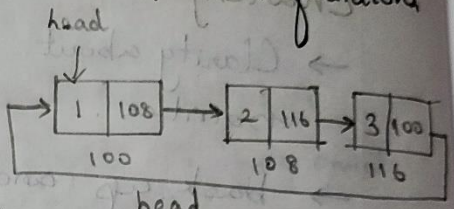
16.11.2022

2. Exercises on Circular Linked List.

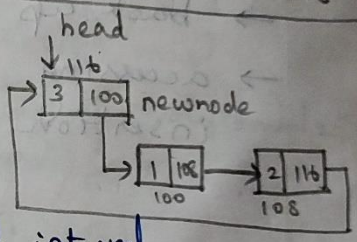
Basic.

Implement a circular linked list abstract data type (ADT) with create, search, print, insert (at the middle, head), delete (at the middle, head) functions

Pseudocode:-




```
struct node
{
    int data;
    struct node *next;
};
```

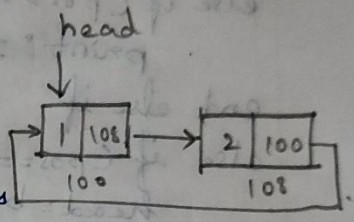



```
Begin_insert()
Input: struct node *head, int val
Output: struct node * head.
newnode = (struct node *) malloc(sizeof(struct node))
if(newnode == NULL)
    print "Cannot allocate memory"
end if
else if(head == NULL)
    newnode -> data = val
    head = newnode
    newnode -> next = head
    print "Node inserted successfully..."
end else if
```

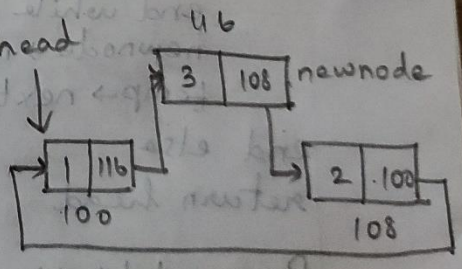
Ex -no : 2
Date: 16-11-22

Name: M.Rohith
3122 21 5001 085

```
else
    newnode -> data = val
    while (temp -> next != head)
        temp = temp -> next
    end while
    temp -> next = newnode
    newnode -> next = head
    head = newnode
end else
return head
```




```
Create()
Input: head, num_nodes
Output: head
i=0, val=0
while (i<num_nodes)
    INPUT val
    head=begin_insert (head, val)
    i = i+1
end while
return head
```



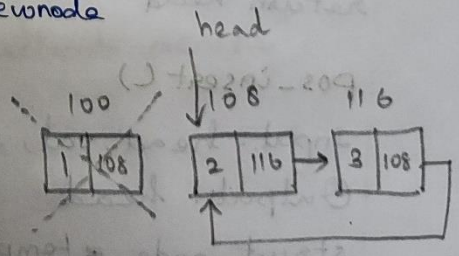

```
pos_insert()
Input: head, val, pos
Output: head
struct node *temp, *check = head, *newnode
newnode = (struct node *) malloc (sizeof (struct node));
newnode -> data = val
l = 0, i = 0
```

Ex -no : 2
Date: 16-11-22

Name: M.Rohith
3122 21 5001 085

```
while (check -> next != head)
    l=l+1;
    check=check -> next
end while
if (head==NULL)
    print "List is empty"
end if
else if (pos > l+1)
    print "Invalid position...."
end else if
else if (pos==0)
    head = begin_insert(head, val)
end else if
else
    temp = head
    while (i< pos-1)
        temp = temp->next
        i=i+1
    end while
    newnode->next = temp->next
    temp-> next = newnode
end else
return head
```

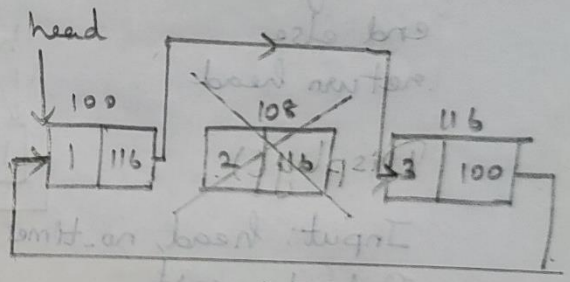
Begin_delete()
Input: head
Output: head



Ex -no : 2
Date: 16-11-22

Name: M.Rohith
3122 21 5001 085

```
if (temp == NULL)
    print "List is empty"
end if
else if (temp→next == temp)
    free (temp)
    head = NULL
end else if
else
    while (temp→next != head)
        temp = temp→next
    temp→next = head→next
    free (head)
    head = temp→next
end else
return head
```



Delete pos()

Input : head, pos

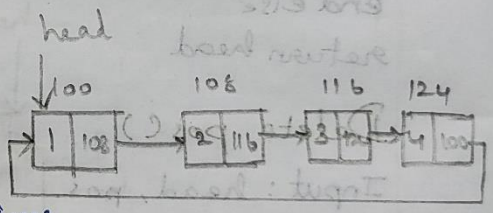
Output: head

```
struct node *temp1, *temp2, *check = head
l = 0, i = 0
if (head == NULL)
    return head
end if
while (check→next != head)
    check = check→next
    l = l + 1
end while
if (pos > l)
    print "Invalid"
end if
```

Ex -no : 2
Date: 16-11-22

Name: M.Rohith
3122 21 5001 085

```
else if (pos==0)
    head = begin_delete(head)
end else if
else
    temp1 = head
    while (temp1→next != head)
        temp2 = temp1
        temp1 = temp1→next
        if (i==pos-1)
            break
        end if
        i = i+1
    end while
    temp2→next = temp1→next
    free(temp1)
end else
return head
```

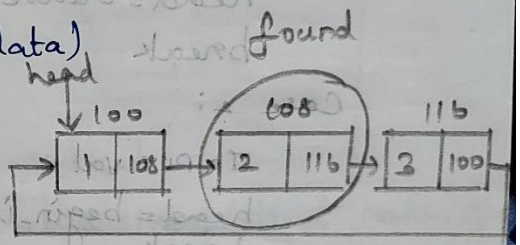


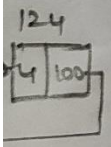

```
Display().
Input: head, no_times
Output: void
struct node *ptr = head
if (ptr == NULL)
    print "Nothing to print..."
end if
else
    for(int i=0; i<no_times; i++)
        ptr = head
        while (ptr→next != head)
            print the Datas
            ptr = ptr→next
        end while
    end for
end else
```

Ex -no : 2
Date: 16-11-22

Name: M.Rohith
3122 21 5001 085

```
Search()
Input: head, val
Output: void
struct node *temp=head
i=0, count=0
if (temp==NULL)
    print "List is Empty"
end if
else
    if (temp->next == head)
        if (val==temp->data)
            count++
        end if
    end if
    else
        if (val==temp->data)
            count++
        end if
        i++
        temp=temp->next
        while (temp!=head
            if (val==temp->data)
                count=count+1
            end if
            i=i+1
            temp=temp->next
        end while
    end else
    if (count=0)
        print " No. of times value found"
    end if
end else
```





Ex -no : 2
Date: 16-11-22

Name: M.Rohith
3122 21 5001 085

Main()

Variables: choice=0, val=0, pos=0
struct node *head = NULL

Output: void.

```
INPUT choice
switch (choice)
case 1:
    head = NULL
    INPUT val
    head = create(head, val)
    break
case 2:
    INPUT val
    head = begin_insert(head, val)
    break
case 3:
    INPUT val
    INPUT pos
    head = pos_insert(head, val, pos)
    break
case 4:
    head = begin_delete(head)
    break
case 5:
    INPUT pos
    head = delete_pos(head, pos)
    break
case 6:
    INPUT val
    Search(head, val)
case 7:
    INPUT val
    display(head, val)
case 8:
    exit
end switch
```

Ex -no : 2
Date: 16-11-22

Name: M.Rohith
3122 21 5001 085

Program Code:

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

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

struct node* begin_insert(struct node *head,int val)
{
    struct node *newnode,*temp=head;
    newnode=(struct node*)malloc(sizeof(struct node));
    if(newnode==NULL)
    {
        printf("\nCannot allocate new memory...\n");
    }
    else if(head==NULL)
        {
            newnode->data=val;
            head=newnode;
            newnode->next=head;
            printf("\nNode inserted successfully...\n");
        }
    else
    {
        newnode->data=val;
            while(temp->next!=head)
                    temp=temp->next;
            temp->next=newnode;
        newnode->next=head;
        head=newnode;
        printf("\nNode inserted successfully\n");
    }
    return head;
}

struct node* create(struct node *head,int num_nodes)
{
    int i=0,val=0;
    while(i<num_nodes)
    {
        printf("\nEnter the value to insert: ");
        scanf("%d",&val);
        head=begin_insert(head,val);
        i++;
    }
```

Name: M.Rohith
3122 21 5001 085

```
    return head;
}

struct node* pos_insert(struct node *head,int val,int pos)
{

    struct node *temp,*check=head,*newnode;
    newnode=(struct node *)malloc(sizeof(struct node));
    newnode->data=val;
    int l=0,i=0;
    while(check->next!=head)
    {
        l++;
        check=check->next;
    }
    if(head==NULL)
    {
        printf("\nList is empty...\n");
    }
    else if(pos>l+1)
        printf("\nInvalid position...\n");
    else if(pos==0)
        head=begin_insert(head,val);
    else
    {
        temp=head;
        while(i<pos-1)
        {
            temp=temp->next;
            i++;
        }
        newnode->next=temp->next;
        temp->next=newnode;
        printf("\nNode inserted at the position successfully...\n");
    }
    return head;

}

struct node* begin_delete(struct node* head)
{
    struct node *temp=head;
    if(temp==NULL)
    {
        printf("\nList is empty...\n");
    }
    else if(temp->next==temp)
    {
```

Name: M.Rohith
3122 21 5001 085

```
        free(temp);
        head=NULL;
        printf("\nOnly node in the list is deleted...\n");
    }
    else
    {
        while(temp->next!=head)
                    temp=temp->next;
            temp->next=head->next;
            free(head);
            head=temp->next;
        printf("\nNode deleted successfully...\n");
    }
    return head;
}

struct node* delete_pos(struct node *head,int pos)
{
    struct node *temp1,*temp2,*check=head;
    int l=0,i=0;
    if(head==NULL)
    {
        printf("\nNothing to delete...\n");
        return head;
    }
    while(check->next!=head)
    {
        check=check->next;
        l++;
    }
    if(pos>l)
        printf("\nInvalid position...\n");
    else if(pos==0)
        head=begin_delete(head);
    else
    {
        temp1=head;
        while(temp1->next!=head)
        {
            temp2=temp1;
            temp1=temp1->next;
            if(i==pos-1)
                break;
            i++;
        }
        temp2->next=temp1->next;
        free(temp1);
        printf("\nNode deleted successfully...\n");
```

Name: M.Rohith
3122 21 5001 085

```
    }
        return head;
}

void display(struct node *head,int no_times)
{
    struct node *ptr;
    ptr=head;
    if(ptr==NULL)
    {
        printf("\nNothing to print...\n");
        }
    else
    {
        printf("\nDisplaying values...");
        for(int i=0;i<no_times;i++)
        {
            ptr=head;
            while(ptr->next!=head)
            {
                printf("\nMemory=%p\tData=%d\tNext=%p",ptr,ptr->data,ptr-
>next);
                ptr=ptr->next;
            }
            printf("\nMemory=%p\tData=%d\tNext=%p",ptr,ptr->data,ptr->next);
        }
        printf("\n");
    }
}

void search(struct node *head,int val)
{
    struct node *temp=head;
        int i=0,count=0;
        if(temp==NULL)
            printf("\nList is Empty");
        else
        {
            if(temp->next==head)
            {
                if(val==temp->data)
                    {
                        count++;
                        printf("\nValue %d is found at the index %d",val,i);
                    }
            }
        else
        {
```

Name: M.Rohith
3122 21 5001 085

```
                if(val==temp->data)
                {
                    count++;
                    printf("\nValue %d is found at the index %d",val,i);
                }
        i++;
                temp=temp->next;
                while(temp!=head)
                {
                    if(val==temp->data)
                    {
                            count++;
                            printf("\nValue %d is found at the position
%d",val,i);
                    }
                    i++;
                    temp=temp->next;
                }
        }
        if(count==0)
            printf("\nThe Value %d is not found in the List",val);
        else
            printf("\nThe Value is found %d time(s)",count);
    }
}

void main()
{
    int choice=0,val=0,pos=0;
    struct node *head=NULL;
    do
    {
        printf("\nChoices :\n1.Create a Circularly linked list\n2.Insert at
the head\n3.Insert at the specified position\n4.Delete at the head\n5.Delete
at the specified position\n6.Search\n7.Display\n8.Exit\n");
        printf("\nEnter your choice: ");
        scanf("%d",&choice);
        switch (choice)
        {
            case 1:
                head=NULL;
                printf("\nEnter the no. of nodes to create: ");
                scanf("%d",&val);
                head=create(head,val);
                break;
            case 2:
                printf("\nEnter the value you want to insert at the beginning:
");
```

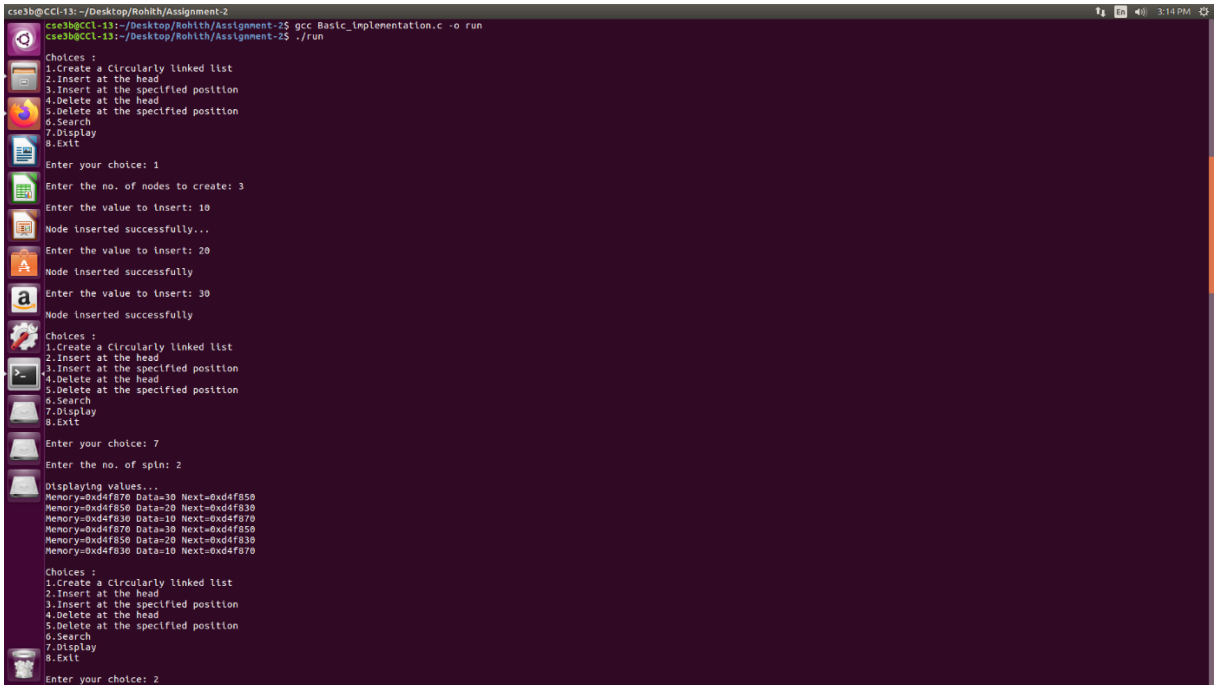
Name: M.Rohith
3122 21 5001 085

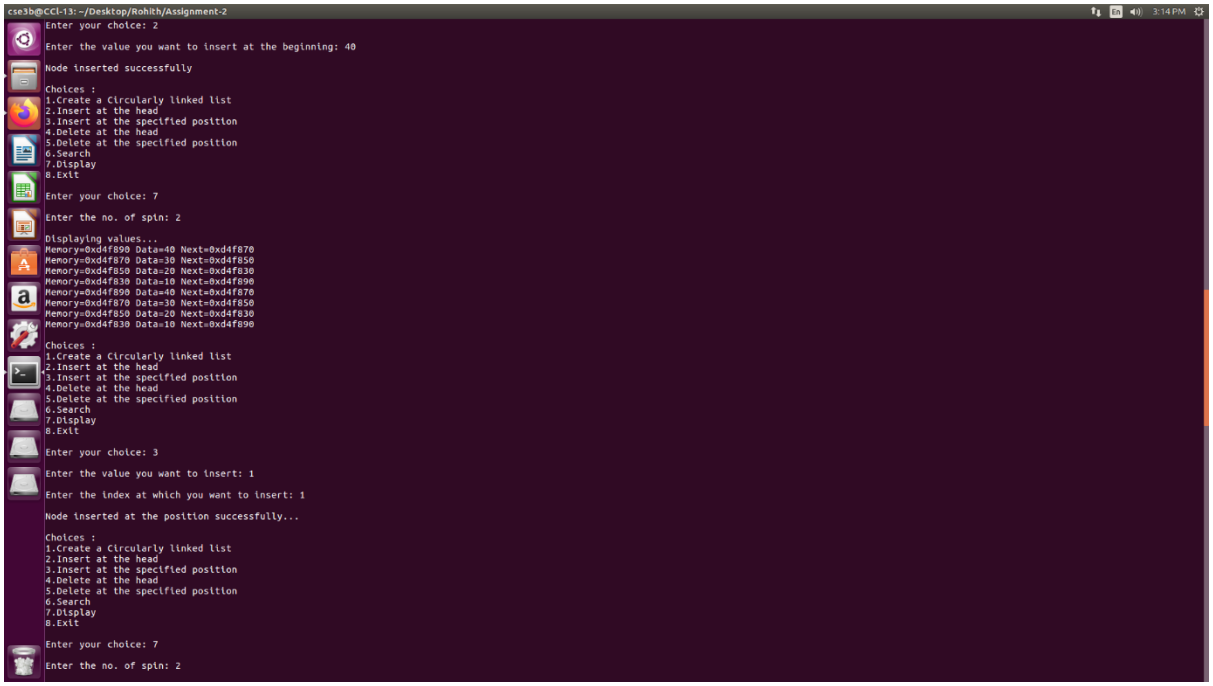
```
                scanf("%d",&val);
                head=begin_insert(head,val);
                break;
            case 3:
                printf("\nEnter the value you want to insert: ");
                scanf("%d",&val);
                printf("\nEnter the index at which you want to insert: ");
                scanf("%d",&pos);
                head=pos_insert(head,val,pos);
                break;
            case 4:
                head=begin_delete(head);
                break;
            case 5:
                printf("\nEnter the index you want to delete: ");
                scanf("%d",&pos);
                head=delete_pos(head,pos);
                break;
            case 6:
                printf("\nEnter the element to search: ");
                scanf("%d",&val);
                search(head,val);
                break;
            case 7:
                printf("\nEnter the no. of spin: ");
                scanf("%d",&val);
                display(head,val);
                break;
            case 8:
                exit(0);
            default:
                printf("\nInvalid choice!!!\n");
        }
    }while(choice!=8);
}
```

Ex -no : 2
Date: 16-11-22

Name: M.Rohith
3122 21 5001 085

Output:



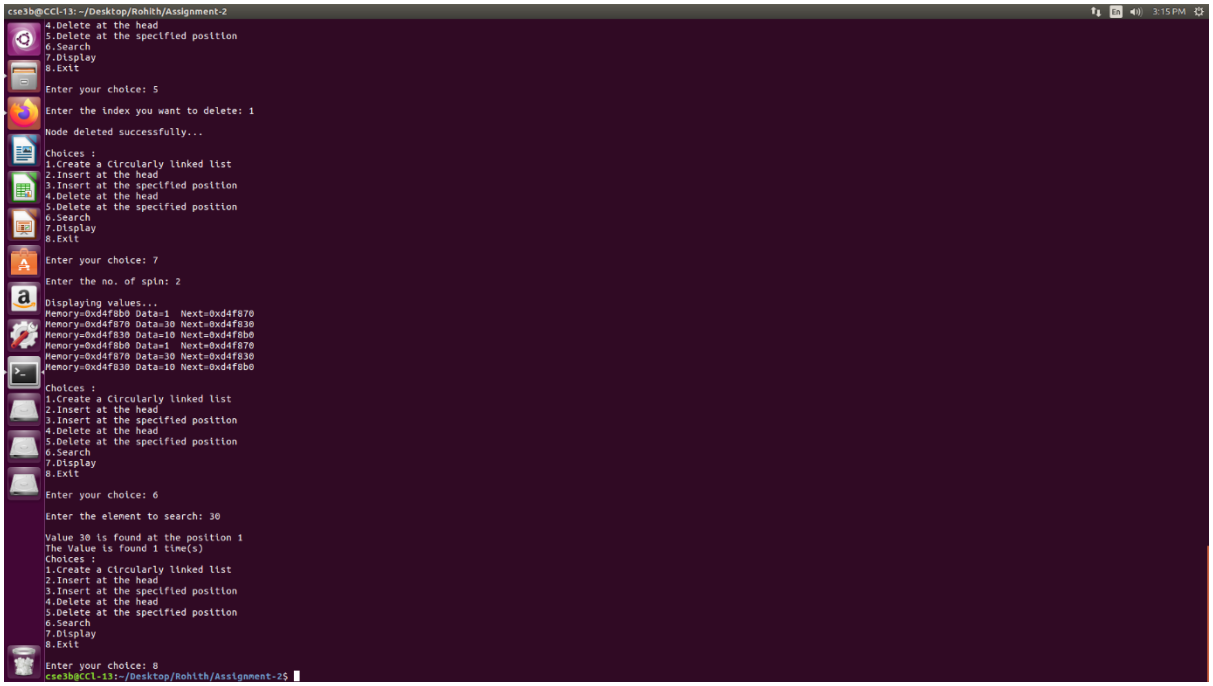


Ex -no : 2
Date: 16-11-22

Name: M.Rohith
3122 21 5001 085

Ex -no : 2
Date: 16-11-22

Name: M.Rohith
3122 21 5001 085



Ex -no : 2
Date: 16-11-22

Name: M.Rohith
3122 21 5001 085

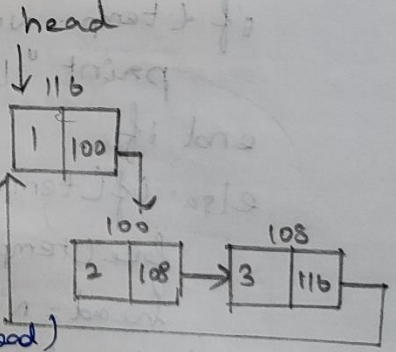
Application

Implement spinning wheel. Rotate the wheel and delete the node at the position in which the wheel stops. Repeat the process until only one node is left

Pseudocode:

Application:-
Implement spinning wheel. Rotate the wheel and delete the node at the position in which the wheel stops. Repeat the process until only one node is left.

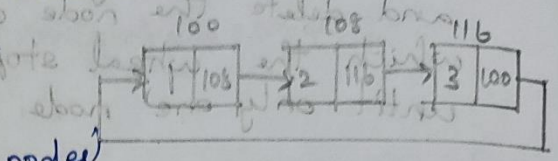
```
struct node
{
    int data
    struct node *next
}

begin_insert()
Input: head, val
Output: head
Struct node *newnode *temp=head
newnode=(struct node*)malloc sizeof(struct node));
if (head==NULL)
    newnode->data=val
    head=newnode
    newnode->next=head
end if
else
    newnode->data=val.
    while (temp->next != head)
        temp=temp->next
    end while
    temp->next=newnode
    newnode->next=head
    head=newnode
end else
return head
```



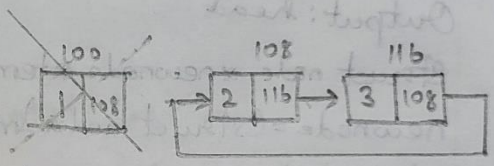
Ex -no : 2
Date: 16-11-22

Name: M.Rohith
3122 21 5001 085

```
Create ()
Input: head, num_nodes
Output: head
i = 0, val = 0
while(i < num_nodes)
    INPUT val
    head = begin_insert(head, val)
    i = i + 1
end while
return head
```



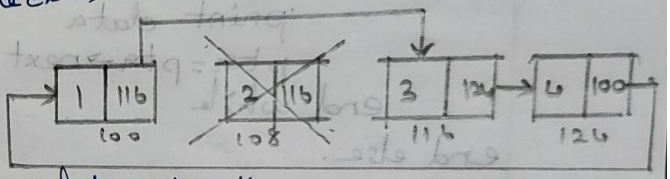

```
begin_delete ()
Input: head
Output: head
if (temp == NULL)
    print "List is empty"
end if
else if(temp -> next == temp)
    free(temp)
    head = NULL
end else if
else
  while (temp -> next != head)
      temp = temp -> next
  end while
  temp -> next = head -> next
  free (head)
  head = temp -> next
end else
return head
```



Ex -no : 2
Date: 16-11-22

Name: M.Rohith
3122 21 5001 085

```
Delete_pos()
Input: struct node *head, int pos
Output : struct node *head
Variables : struct node *temp1, *temp2, *check=head;
            int  l=0, i=0

if (head==NULL)
    return head
end if
while(check->next != head)
    check= check->next
    l=l+1
end while
if (pos > l)
    print "Invalid position"
end if
else if (pos==0)
    head = begin_delete(head)
end else if
else
    temp1 = head
    while(temp1->next != head)
        temp2 = temp1
        temp1 = temp1->next
        if(i == pos-1)
            break
        end if
        i = i+1
    end while
    temp2-> next = temp1-> next
    head= temp1->next
    free(temp1)
end else
return head
```



Ex -no : 2
Date: 16-11-22

Name: M.Rohith
3122 21 5001 085

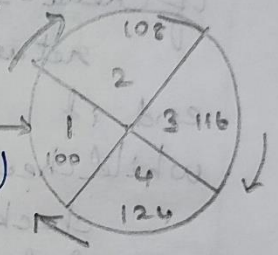
Display()

Input: struct node *head

Output: void

Variables: struct node *ptr = head

```
if (ptr==NULL)
    print "Nothing to print"
end if
else
    ptr=head
    while(ptr->next != head)
        print data
        ptr=ptr->next
    end while
end else.
```



Main().

Variables: struct node *head=NULL, int num, index

```
INPUT num
head = create(head, num)
print the spinning wheel, display(head)
while(num>1)
    if(num==1)
        break
    end if
    index = rand()%(num-1)
    head = delete_pos(head,index)
    display(head)
    num = num-1
end while
print "The data of spinning wheel"
display(head)
```

Ex -no : 2
Date: 16-11-22

Name: M.Rohith
3122 21 5001 085

Program Code:

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

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

struct node* begin_insert(struct node *head,int val)
{
    struct node *newnode,*temp=head;
    newnode=(struct node*)malloc(sizeof(struct node));
    if(newnode==NULL)
    {
        printf("\nCannot allocate new memory...\n");
    }
    else if(head==NULL)
    {
        newnode->data=val;
        head=newnode;
        newnode->next=head;
    }
    else
    {
        newnode->data=val;
        while(temp->next!=head)
            temp=temp->next;
        temp->next=newnode;
        newnode->next=head;
        head=newnode;
    }
    return head;
}

struct node* create(struct node *head,int num_nodes)
{
    int i=0,val=0;
    while(i<num_nodes)
    {
        printf("\nEnter the value to insert: ");
        scanf("%d",&val);
        head=begin_insert(head,val);
        i++;
    }
```

Name: M.Rohith
3122 21 5001 085

```
    return head;
}

struct node* begin_delete(struct node* head)
{
    struct node *temp=head;
    if(temp==NULL)
    {
        printf("\nList is empty...\n");
    }
    else if(temp->next==temp)
    {
        free(temp);
        head=NULL;
        printf("\nOnly node in the list is deleted...\n");
    }
    else
    {
        while(temp->next!=head)
            temp=temp->next;
        temp->next=head->next;
        free(head);
        head=temp->next;
        // printf("\nNode deleted successfully...\n");
        printf("\nSpinning wheel starts at data %d\n",head->data);
    }
    return head;
}

struct node* delete_pos(struct node *head,int pos)
{
    struct node *temp1,*temp2,*check=head;
    int l=0,i=0;
    if(head==NULL)
    {
        printf("\nNothing to delete...\n");
        return head;
    }
    while(check->next!=head)
    {
        check=check->next;
        l++;
    }
    if(pos>l)
        printf("\nInvalid position...\n");
    else if(pos==0)
        head=begin_delete(head);
    else
```

Name: M.Rohith
3122 21 5001 085

```
    {
        temp1=head;
        while(temp1->next!=head)
        {
            temp2=temp1;
            temp1=temp1->next;
            if(i==pos-1)
                break;
            i++;
        }
        temp2->next=temp1->next;
        head=temp1->next;
        free(temp1);
        printf("\nSpinnning wheel starts from data %d\n",head->data);
    }
    return head;
}

void display(struct node *head)
{
    struct node *ptr;
    ptr=head;
    if(ptr==NULL)
    {
        printf("\nNothing to print...\n");
        }
    else
    {
        ptr=head;
        while(ptr->next!=head)
        {
            printf("\nData=%d",ptr->data);
            ptr=ptr->next;
        }
        printf("\nData=%d",ptr->data);
        printf("\n");
    }
}

void main()
{
    struct node *head=NULL;
    int num,index;
    printf("\nEnter the number of nodes you want to insert: ");
    scanf("%d",&num);
    head=create(head,num);
    printf("\n*** AT THE BEGINNING ***\n");
    printf("Spinning wheel data :");
```

Name: M.Rohith
3122 21 5001 085

```
    display(head);
    while(num>1)
    {
        printf("\n*********************************\n");
        printf("\nSpinning the wheel...\n");
        if(num==1)
            break;
        index=rand()%(num-1);
        printf("\nThe wheel stopped at index %d\n",index);
        head=delete_pos(head,index);
        printf("\nSpinning wheel data :\n");
        display(head);
        num--;
        printf("\n*********************************\n");
    }
    printf("\n*** AT THE END ***\n");
    printf("Spinning wheel data :");
    display(head);
    printf("\n");
}
```

Ex -no : 2 Name: M.Rohith
Date: 16-11-22 3122 21 5001 085

Output:

```
Enter the number of nodes you want to insert: 5

Enter the value to insert: 10

Enter the value to insert: 20

Enter the value to insert: 30

Enter the value to insert: 40

Enter the value to insert: 50

*** AT THE BEGINNING ***
Spinning wheel data :
Data=50
Data=40
Data=30
Data=20
Data=10
```

```
**********************************

Spinning the wheel...

The wheel stopped at index 1

Spinnning wheel starts from data 30

Spinning wheel data :

Data=30
Data=20
Data=10
Data=50

**********************************

**********************************

Spinning the wheel...

The wheel stopped at index 2

Spinnning wheel starts from data 50

Spinning wheel data :

Data=50
Data=30
Data=20

**********************************
```

Ex -no : 2 Name: M.Rohith
Date: 16-11-22 3122 21 5001 085

```
**********************************

Spinning the wheel...

The wheel stopped at index 0

Spinning wheel starts at data 30

Spinning wheel data :

Data=30
Data=20

**********************************

**********************************

Spinning the wheel...

The wheel stopped at index 0

Spinning wheel starts at data 20

Spinning wheel data :

Data=20

**********************************

*** AT THE END ***
Spinning wheel data :
Data=20
```

Results:

Thus C program using Circular linked List in data structures as well as the spinning wheel using circular linked list have been implemented and executed successfully.