Ex no.:3 Name: M. Rohith
Date: 16-11-2022 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 3: Exercises on doubly linked list

Aim:

To implement C program by using Doubly Linked List.

Basic

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

Ex no.:3 Name: M. Rohith
Date: 16-11-2022 3122 21 5001 085

Pseudocode:

16.11.22.

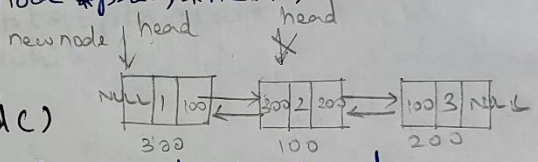
3. Exercises on Doubly linked list:-

Basic.

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

Algorithm:-

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



Insert_head()

Input: struct node *head, int val

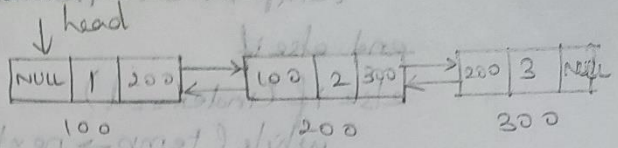
Output: head.

```
Struct node *newnode = (struct node*)malloc(sizeof(struct node));
newnode -> data = val
if (head == NULL)
    newnode -> next = NULL
    newnode -> prev = NULL
    head = newnode
end if
else
    newnode -> next = head
    newnode -> prev = NULL
    head -> prev = newnode;
    head = newnode
end else
return head
```

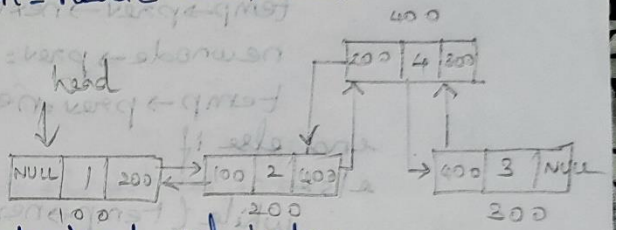
Ex no.:3
Date: 16-11-2022

Name: M. Rohith
3122 21 5001 085

Create ()
Input: head, num_nodes
Output: head



val = 0
head = NULL
for i = 0 to num_nodes
    INPUT val
    head = insert_head(head, val)
end for
return head



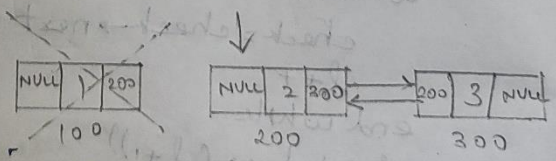
Insert_pos ()
Input: struct node *head, val, index
Output: head

struct node *newnode, *temp = head, *check = head
newnode = (struct node*) malloc(sizeof(struct node))
newnode → data = val
if (index == 0)
    head = insert_head(head, val)
end if
else
    while(check → next != NULL)
        check = check → next
        l++
    end while
    if(index > (l+1))
        print "Invalid
    end if
    else if(index == (l+1))
        while (temp → next != NULL)
            temp = temp → next
        end while

Ex no.:3
Date: 16-11-2022

Name: M. Rohith
3122 21 5001 085

```
        newnode->next = temp
        temp->next = newnode;
    end else if
    else if (index == 1)
        while (temp->next != NULL)
            temp = temp->next;
        end while
        newnode->next = temp
        temp->prev->next = newnode
        newnode->prev = temp->prev
        temp->prev = newnode
    end else if
    else
        while (temp->next != NULL)
            if (i == index-1)
                break
            end if
            temp = temp->next;
            i++
        end while
        newnode->next = temp->next
        newnode->prev = temp
        temp->next->prev = newnode
        temp->next = newnode.
    end else
return head.
```




```
Del_head ()
Input: head
Output: head.
struct node *temp = head
if (head == NULL)
    print "List"
end if.
```

Ex no.:3
Date: 16-11-2022

Name: M. Rohith
3122 21 5001 085

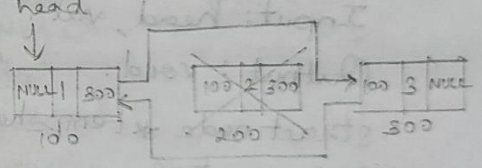
```
else if (head -> next == NULL)
        free(temp)
        head = NULL
end else if
else
        head = temp -> next
        head -> prev = NULL;
        free(temp)
end else
return head.
```

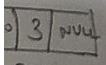
Del_pos()

Input: struct node *head, int index

Output: head

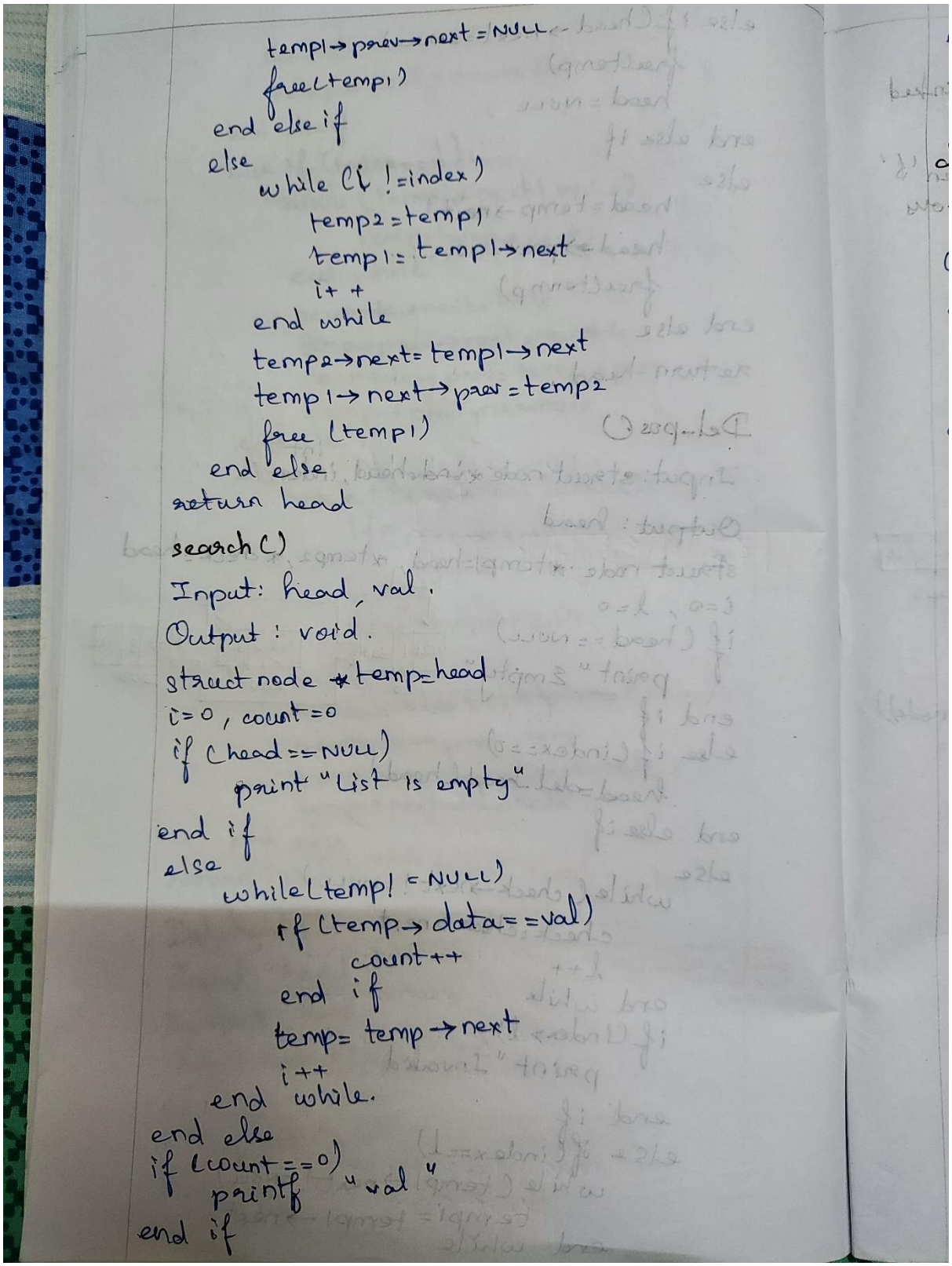
```
struct node *temp1=head, *temp2, *check = head
i=0, l=0
if (head == NULL)
    print " Empty"
end if
else if (index==0)
    head=del_head(head)
end else if
else
    while (check->next != NULL)
        check=check->next
        l++
    end while
    if (index > l)
        print "Invalid"
    end if
    else if (index==l)
        while (temp1->next!=NULL)
            temp1= temp1->next
        end while
```





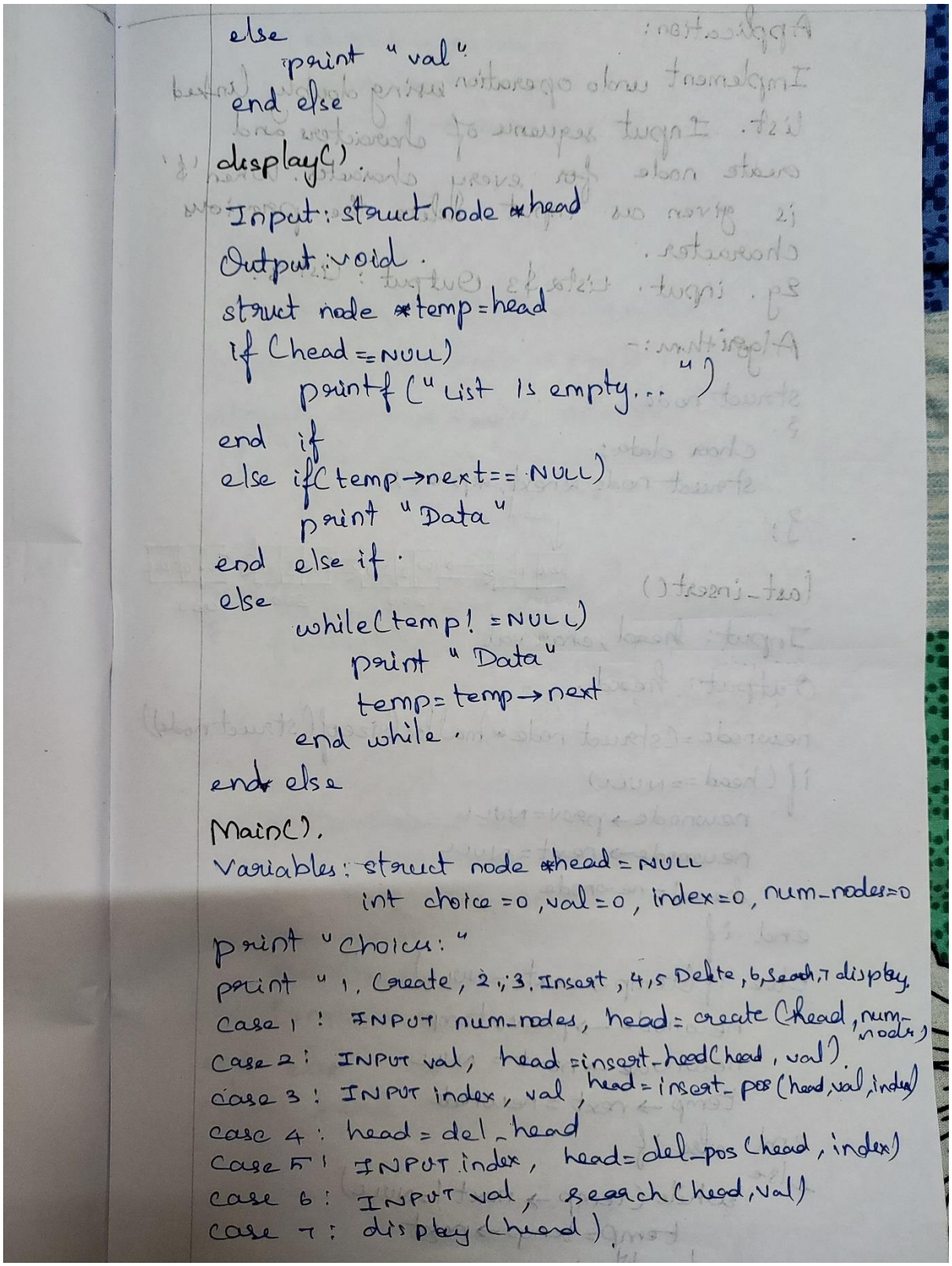
Ex no.:3
Date: 16-11-2022

Name: M. Rohith
3122 21 5001 085



```
        temp1->prev->next = NULL
        free(temp1)
    end else if
    else
        while (i != index)
            temp2 = temp1
            temp1 = temp1->next
            i++
        end while
        temp2->next = temp1->next
        temp1->next->prev = temp2
        free (temp1)
    end else
return head
search()
Input: head, val.
Output : void.
struct node *temp=head
i=0, count=0
if (head==NULL)
    print "List is empty"
end if
else
    while(temp! = NULL)
        if (temp->data==val)
            count++
        end if
        temp= temp->next
        i++
    end while.
end else
if (count==0)
    printf "val"
end if
```

Name: M. Rohith
3122 21 5001 085



```
else
    print "val"
end else
display().
Input: struct node *head
Output: void.
struct node *temp=head
if (head==NULL)
    printf("List is empty...")
end if
else if(temp->next== NULL)
    print "Data"
end else if.
else
    while(temp! =NULL)
        print "Data"
        temp= temp->next
    end while.
end else
Main().
Variables: struct node *head=NULL
        int choice=0, val=0, index=0, num_nodes=0
print "Choice:"
print "1. Create, 2.;3. Insert, 4,5 Delete, 6, Search, 7 display.
case 1 : INPUT num_nodes, head= create(head, num_nodes)
case 2: INPUT val; head =insert_head(head, val).
case 3 : INPUT index, val, head=insert_pos(head,val,index)
case 4 : head = del_head
case 5 : INPUT index, head=del_pos(head, index)
case 6 : INPUT val, search(head,val)
case 7 : display(head).
```

Ex no.:3
Date: 16-11-2022

Name: M. Rohith
3122 21 5001 085

Program code:

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

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

struct node* insert_head(struct node* head,int val)
{
    struct node *newnode=(struct node*)malloc(sizeof(struct node));
    newnode->data=val;
    if(head==NULL)
    {
        newnode->next=NULL;
        newnode->prev=NULL;
        head=newnode;
    }
    else
    {
        newnode->next=head;
        newnode->prev=NULL;
        head->prev=newnode;
        head=newnode;
    }
    printf("\nNode inserted successfully...\n");
    return head;
}

struct node* create(struct node* head,int num_nodes)
{
    int val=0;
    head=NULL;
    for(int i=0;i<num_nodes;i++)
    {
        printf("\nEnter the data: ");
        scanf("%d",&val);
        head=insert_head(head,val);
    }
    return head;
}

struct node* insert_pos(struct node *head,int val,int index)
{
```

Name: M. Rohith
3122 21 5001 085

```
    int l=0,i=0;
    struct node *newnode,*temp=head,*check=head;
    newnode=(struct node*)malloc(sizeof(struct node));
    newnode->data=val;
    if(index==0)
        head=insert_head(head,val);
    else
    {
        while(check->next!=NULL)
        {
            check=check->next;
            l++;
        }
        if(index>(l+1))
            printf("\nInvalid position...\n");
        else if(index==(l+1))
        {
            while(temp->next!=NULL)
                temp=temp->next;
            newnode->next=NULL;
            newnode->prev=temp;
            temp->next=newnode;
            printf("\nNode inserted succcessfully...\n");
        }
        else if(index==l)
        {
            while(temp->next!=NULL)
                temp=temp->next;
            newnode->next=temp;
            temp->prev->next=newnode;
            newnode->prev=temp->prev;
            temp->prev=newnode;
            printf("\nNode inserted succcessfully...\n");
        }
        else
        {
            while(temp->next!=NULL)
            {
                if(i==index-1)
                    break;
                temp=temp->next;
                i++;
            }
            newnode->next=temp->next;
            newnode->prev=temp;
            temp->next->prev=newnode;
            temp->next=newnode;
            printf("\nNode inserted succcessfully...\n");
```

Name: M. Rohith
3122 21 5001 085

```
        }
    }
    return head;
}

struct node* del_head(struct node* head)
{
    struct node *temp=head;
    if(head==NULL)
        printf("\nList is empty...\n");
    else if(head->next==NULL)
    {
        free(temp);
        head=NULL;
        printf("\nNode deleted successfully...\n");
    }
    else
    {
        head=temp->next;
        head->prev=NULL;
        free(temp);
        printf("\nNode deleted successfully...\n");
    }
    return head;
}

struct node* del_pos(struct node *head,int index)
{
    struct node *temp1=head,*temp2,*check=head;
    int i=0,l=0;
    if(head==NULL)
        printf("\nList is empty...\n");
    else if(index==0)
        head=del_head(head);
    else
    {
        while(check->next!=NULL)
        {
            check=check->next;
            l++;
        }
        if(index>l)
            printf("\nInvalid index...\n");
        else if(index==l)
        {
            while(temp1->next!=NULL)
                temp1=temp1->next;
            temp1->prev->next=NULL;
```

Name: M. Rohith
3122 21 5001 085

```
            free(temp1);
            printf("\nNode deleted successfully...\n");
        }
        else
        {
            while(i!=index)
            {
                temp2=temp1;
                temp1=temp1->next;
                i++;
            }
            temp2->next=temp1->next;
            temp1->next->prev=temp2;
            free(temp1);
            printf("\nNode deleted successfully...\n");
        }
    }
    return head;
}

void search(struct node *head,int val)
{
    struct node *temp=head;
    int i=0,count=0;
    if(head==NULL)
        printf("\nList is emty...\n");
    else
    {
        while(temp!=NULL)
        {
            if(temp->data==val)
            {
                printf("\n%d found at index %d",val,i);
                count++;
            }
            temp=temp->next;
            i++;
        }
    }
    if(count==0)
        printf("\n%d found %d times\n",val,count);
    else
        printf("\n%d found %d times\n",val,count);
}

void display(struct node *head)
{
    struct node *temp=head;
```

Name: M. Rohith
3122 21 5001 085

```
    printf("\nDisplaying values...\n");
    if(head==NULL)
        printf("\nList is empty...\n");
    else if(temp->next==NULL)
    {
        printf("\nMemory=%p\tPrev=%p\tData=%d\tNext=%p\n",temp,temp-
>prev,temp->data,temp->next);
    }
    else
    {
        while(temp!=NULL)
        {
            printf("\nMemory=%p\tPrev=%p\tData=%d\tNext=%p\n",temp,temp-
>prev,temp->data,temp->next);
            temp=temp->next;
        }
    }
    printf("\n");
}

void main()
{
    struct node *head=NULL;
    int choice=0,val=0,index=0,num_nodes=0;
    do
    {
        printf("\nChoices:\n");
        printf("\n1.Create\n2.Insert at head\n3.Insert at the
position\n4.Delete at the head\n5.Delete at the
position\n6.Search\n7.Display\n8.Exit\n");
        printf("\nEnter your choice: ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
                printf("\nEnter the number of nodes: ");
                scanf("%d",&num_nodes);
                head=create(head,num_nodes);
                break;
            case 2:
                printf("\nEnter the data: ");
                scanf("%d",&val);
                head=insert_head(head,val);
                break;
            case 3:
                printf("\nEnter the index to insert: ");
                scanf("%d",&index);
                printf("\nEnter the data: ");
```

Name: M. Rohith
3122 21 5001 085

```
                scanf("%d",&val);
                head=insert_pos(head,val,index);
                break;
            case 4:
                head=del_head(head);
                break;
            case 5:
                printf("\nEnter the index to delete: ");
                scanf("%d",&index);
                head=del_pos(head,index);
                break;
            case 6:
                printf("\nEnter the value to be searched: ");
                scanf("%d",&val);
                search(head,val);
                break;
            case 7:
                display(head);
                break;
            case 8:
                exit(0);
            default:
                printf("\nInvalid choice...\n");
        }
    } while (choice!=8);
}
```

Ex no.:3 Name: M. Rohith
Date: 16-11-2022 3122 21 5001 085

Output:

```
PS C:\Rohith\Backup\Desktop\SEM 3\Data Structures in C\Assignment-3> gcc Basic_Implementation.c -o run
PS C:\Rohith\Backup\Desktop\SEM 3\Data Structures in C\Assignment-3> ./run

Choices:

1.Create
2.Insert at head
3.Insert at the position
4.Delete at the head
5.Delete at the position
6.Search
7.Display
8.Exit

Enter your choice: 1

Enter the number of nodes: 3

Enter the data: 10

Node inserted successfully...

Enter the data: 20

Node inserted successfully...

Enter the data: 30

Node inserted successfully...

Choices:

1.Create
2.Insert at head
3.Insert at the position
4.Delete at the head
5.Delete at the position
6.Search
```

```
7.Display
8.Exit

Enter your choice: 2

Enter the data: 40

Node inserted successfully...

Choices:

1.Create
2.Insert at head
3.Insert at the position
4.Delete at the head
5.Delete at the position
6.Search
7.Display
8.Exit

Enter your choice: 7

Displaying values...

Memory=00C80E00 Prev=00000000   Data=40 Next=00C80DE8

Memory=00C80DE8 Prev=00C80E00   Data=30 Next=00C813C0

Memory=00C813C0 Prev=00C80DE8   Data=20 Next=00C813A8

Memory=00C813A8 Prev=00C813C0   Data=10 Next=00000000
```

Ex no.:3 Name: M. Rohith
Date: 16-11-2022 3122 21 5001 085

```
Choices:

1.Create
2.Insert at head
3.Insert at the position
4.Delete at the head
5.Delete at the position
6.Search
7.Display
8.Exit

Enter your choice: 3

Enter the index to insert: 1

Enter the data: 35

Node inserted succcessfully...

Choices:

1.Create
2.Insert at head
3.Insert at the position
4.Delete at the head
5.Delete at the position
6.Search
7.Display
8.Exit
```

```
Enter your choice: 7

Displaying values...

Memory=00C80E00 Prev=00000000   Data=40 Next=00C80E18

Memory=00C80E18 Prev=00C80E00   Data=35 Next=00C80DE8

Memory=00C80DE8 Prev=00C80E18   Data=30 Next=00C813C0

Memory=00C813C0 Prev=00C80DE8   Data=20 Next=00C813A8

Memory=00C813A8 Prev=00C813C0   Data=10 Next=00000000


Choices:

1.Create
2.Insert at head
3.Insert at the position
4.Delete at the head
5.Delete at the position
6.Search
7.Display
8.Exit

Enter your choice: 4

Node deleted successfully...
```

Ex no.:3 Name: M. Rohith
Date: 16-11-2022 3122 21 5001 085

```
Choices:

1.Create
2.Insert at head
3.Insert at the position
4.Delete at the head
5.Delete at the position
6.Search
7.Display
8.Exit

Enter your choice: 7

Displaying values...

Memory=00C80E18 Prev=00000000   Data=35 Next=00C80DE8

Memory=00C80DE8 Prev=00C80E18   Data=30 Next=00C813C0

Memory=00C813C0 Prev=00C80DE8   Data=20 Next=00C813A8

Memory=00C813A8 Prev=00C813C0   Data=10 Next=00000000


Choices:

1.Create
2.Insert at head
3.Insert at the position
4.Delete at the head
5.Delete at the position
6.Search
7.Display
8.Exit
```

```
Enter your choice: 5

Enter the index to delete: 3

Node deleted successfully...

Choices:

1.Create
2.Insert at head
3.Insert at the position
4.Delete at the head
5.Delete at the position
6.Search
7.Display
8.Exit

Enter your choice: 7

Displaying values...

Memory=00C80E18 Prev=00000000   Data=35 Next=00C80DE8

Memory=00C80DE8 Prev=00C80E18   Data=30 Next=00C813C0

Memory=00C813C0 Prev=00C80DE8   Data=20 Next=00000000
```

Ex no.:3 Name: M. Rohith
Date: 16-11-2022 3122 21 5001 085

```
Choices:

1.Create
2.Insert at head
3.Insert at the position
4.Delete at the head
5.Delete at the position
6.Search
7.Display
8.Exit

Enter your choice: 6

Enter the value to be searched: 30

30 found at index 1
30 found 1 times

Choices:

1.Create
2.Insert at head
3.Insert at the position
4.Delete at the head
5.Delete at the position
6.Search
7.Display
8.Exit

Enter your choice: 8
PS C:\Rohith\Backup\Desktop\SEM 3\Data Structures in C\Assignment-3>
```

Ex no.:3 Name: M. Rohith
Date: 16-11-2022 3122 21 5001 085

Application

Implement Undo operation using doubly linked list. Input sequence of characters and create node for every character. When '$' is given as input delete the previous character.

Eg: Input: List2$3

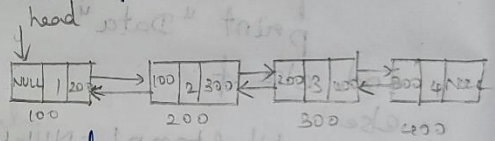
Output:List3

Pseudocode:

Application:
Implement undo operation using doubly linked list. Input sequence of characters and create node for every character. When '$' is given as input delete the previous character.
Eg. input. List2$3 Output : List3.

Algorithm:-

```
struct node
{
    char data;
    struct node *next, *prev;
};
```




```
last_insert()
Input: head, char val
Output: head.
newnode=(struct node *)malloc(sizeof(struct node))
if (head==NULL)
    newnode->prev=NULL
    newnode->next=NULL
    head=newnode
end if
else if (temp->next==NULL)
    newnode->prev=temp
    newnode->next = NULL
    temp->next =newnode
end else if
else
    while (temp->next!=NULL)
        temp=temp->next
    end while
```

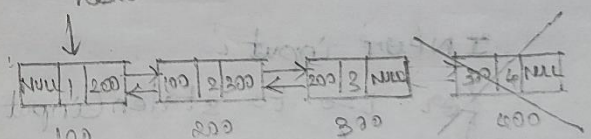
Ex no.:3
Date: 16-11-2022

Name: M. Rohith
3122 21 5001 085

```
        newnode->prev=temp
        temp->next=newnode
        newnode->next=NULL
end else
return head.
```

last_delete()

```
Input: struct node *temp1, *temp2
Output: struct node * head
struct node * temp1, *temp2
if (head==NULL)
    print "List is empty"
end if
else if (temp1->next==NULL)
{   free(head);
    head=NULL
}end else if
else
    temp1=head
    while (temp1->next!=NULL)
        temp2=temp1
        temp1=temp1->next
    end while
    free (temp1)
    temp2->next=NULL
end else
return head
```



Ex no.:3
Date: 16-11-2022

Name: M. Rohith
3122 21 5001 085

```
Display().
Input: struct node *head
Output: void
struct node *temp=head
if (head==NULL)
    print" Empty list "
end if
else
    while(temp!=NULL)
        print temp->data
        temp=temp->next
    end while.
end else

Main()
Variables: struct node *head=NULL
           char input[no]

INPUT input
for i=0 to strlen(input)
    if (input[i]=='$')
        head=last_delete(head)
    end if
    else
        head=last_insert(head, input[i])
    end else
end for
display(head)
```

Ex no.:3
Date: 16-11-2022

Name: M. Rohith
3122 21 5001 085

Program code:

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

struct node
{
    char data;
    struct node *next,*prev;
};

struct node* last_insert(struct node *head,char val)
{
    struct node *temp=head,*newnode;
    newnode=(struct node*)malloc(sizeof(struct node));
    newnode->data=val;
    if(head==NULL)
    {
        newnode->next=NULL;
        newnode->prev=NULL;
        head=newnode;
    }
    else if(temp->next==NULL)
    {
        newnode->prev=temp;
        newnode->next=NULL;
        temp->next=newnode;
    }
    else
    {
        while(temp->next!=NULL)
            temp=temp->next;
        newnode->prev=temp;
        temp->next=newnode;
        newnode->next=NULL;
    }
    // printf("\nNode inserted at the last successfully...\n");
    return head;
}

struct node* last_delete(struct node *head)
{
    struct node *temp1,*temp2;
    if(head==NULL)
        printf("\nList is empty\n");
    else if(temp1->next==NULL)
```

Name: M. Rohith
3122 21 5001 085

```
    {
        free(head);
        head=NULL;
        // printf("\nNode deleted successfully...\n");
    }
    else
    {
        temp1=head;
        while(temp1->next!=NULL)
        {
            temp2=temp1;
            temp1=temp1->next;
        }
        free(temp1);
        temp2->next=NULL;
        // printf("\nNode deleted successfully...\n");
    }
    return head;
}

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

void main()
{
    struct node *head=NULL;
    char input[50];
    printf("\nEnter the input: ");
    gets(input);

    for(int i=0;i<strlen(input);i++)
    {
        if(input[i]=='$')
            head=last_delete(head);
```

```
        else
            head=last_insert(head,input[i]);
    }
    display(head);
}
```

Output:

```
PS C:\Rohith\Backup\Desktop\SEM 3\Data Structures in C\Assignment-3> gcc Application.c -o run
PS C:\Rohith\Backup\Desktop\SEM 3\Data Structures in C\Assignment-3> ./run

Enter the input: List2$3

Displaying values...
L i s t 3
PS C:\Rohith\Backup\Desktop\SEM 3\Data Structures in C\Assignment-3>
```

Results:

Thus C program using Doubly Linked List in data structures as well as the program to implement undo operation using Doubly Linked List have been implemented and executed successfully.