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.

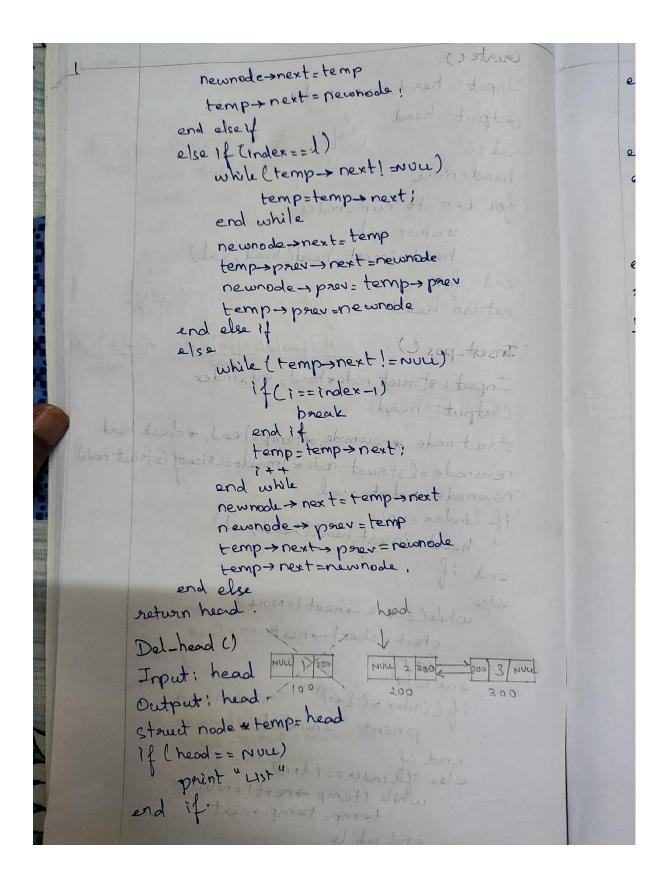
Basic

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

Pseudocode:

16.11.22.		
(b, (l\22.	Basic. Basic. Implement a doubly linked list abstract Implement a doubly linked list abstract data type (ADT) with create, point, search insert (at the middle, head), delete (at the middle, head) functions.	1,
9000 2000 2000	Algorithm:	l
	struct node sint data;	9
	3! new node head head	
	Insert-head () Null 100 200 200 3 Nac 200 200 200 200 200 200 200 200 200 20	0
	Output: head. Struct rode «neuonode = (struct rode *) malloc (size of contract) (struct rode)	2
	if (head == NULL)	
	newnode -> next = NULL newnode -> prev = NULL head = newnode	
	end if else hewnode = next=head	The same of the sa
	head ->prev= newnode; head = newnode	The second secon
3	end else return head	the same of the same of the same of

	Courte () and of strange about
	Input: head, hum-nodes
_	Vanda bas
2 earch	
	val =0 (200 (4x00)00 grant) de 1200 300
te	head=NULL
	for i=0 to num-nodes
	1000
	head = insert-head (head, val)
	end forgedment suggestioners too 4/20)
	return head grand of
	Insert-pos () NULL 1 200 2 400 3 Nyle 200
	Input: et ruet node * head, val, index
	Input: et sua moderna
+	Output: head
	stand node mnevonode, attemp= head, * check=head
	rewnode = (struct rode *) malloc(size of (struct node)
	rewrode - data=val
=(sizeof	1.0 A. La A.
ruit nodely	head = insert-head (head, val)
	end if abonainet son egner
	else 1 1 1 1 1 1 1 1 1
	else while Ccheck > next != NULL)
	check= check-next () book-loc
	1++
	end while
	if Cindax > Cl+1)
	point Invalid
	end if else iftinder == (d+1))
	else iftinder == (tti)
	while (temp > next!= NULL)
	end while
	Era while



else if Chead > next==NULL) fractemp) head = NULL end alse if else head => prev = NULL; fracternp) end else neturn head. Del-pos () Input: struct node xindehead, index Output: head struct rode ** temp!=head, ** xtemp2, ** check=head i=0, 1-9
tractemp) head = NULL and also if also head = temp -> next head => poor = NULL; freetern p) end also neturn head. Del-pos () Input: etouct node * ind-chead, index Or head: head
head = NULL end alse If else head = temp > next head > prov = NULL; freetern p) end else neturn head. Del-pas () Input: struct node * indehead, index
end also if else head = temp = next head => prev = noul; freeltennp) end else neturn head. Del-pos C) Input: struct node x indehead, index On location head.
head = temp > next head => prov = noul; freeltennp) end else neturn head. Del-pos () Input: etract node xind chead, index
head = temp > next head => prov = noul; freeltennp) end else neturn head. Del-pos () Input: etract node xind chead, index
head >> prev = NOLL; freectern p) end else neturn head. Del-pos () Input: et ract node x indohead, index
pet-pos C) Input: et suct node * index
Del-pos C) Input: et auct node * index
Del-pos C) Input: et auct node * indehead, index
Del-pos () Input: etauct node xindohead, index
Input: stauct node * indehead, index
a 1 t. head
Output: head
1 standshood extemps, excheck = head
stacet node * Camping
i=0, l=0
if (head == NULL) point " 2 mpty" 100 2 300 300 300
paint 2 mpty
end if
else if Lindex == 0) head=del_head (head)
head -del sund control
end else if
else while (check - next!= NULL)
check=check > next
3 mul
and while
if Cinders 1)
paint "Invalid
end if
else if (index==1)
while (temp) - next (= nucl)
end while templ -> next

1 1 1 3 3 621-	
templ -> porev -> next = Nucle - band of seed	
face (temps)	bestri
and else if	
alse all 1	٥١١١٥
faceltempi) end else if else while (i !=index) temp2 = temp1	A UT
temp2=temp1	8.0
temp1= temp1 > next	(
it + end while	
end while temps > next = templ > next town 1 > nont > nont = temp2	
temps > next = templ > next temps > next > pass = temp2	
free (temp1) () 289-15-1	
end else band bank about the toget	
board: tright	
end else sead based to the search C) Input: head val. Output: void.	1 64
Input: head, val.	
struct node * temp=head igns "topog i=0, count=0	
i=0, count=0	Malabap
if chead == NULL) paint "List is empty"	
paint "List is empty"	
end if	
else while Ltemp! = NULL) of Ctemp > data = = val)	
rf Ctemp > data = = vai	
end if	
· · · · · · · · · · · · · · · · · · ·	
temp= temp -> next end while.	
end unble.	
end alle	
if count == 0)	
end else if count == 0) prints	
end if and algorithm bear	

bette point "val"

bette prime nothere of olone transformit

tend else prime nothere of translated to the stars

to display ()

in display () Input: stauct rode whend in month Output void. struct node *temp=head

if Chead == NULL)

point f (" List 1s empty...) end else if Ctemp-next == NULL) print "Data" end else if. else while (temp! = NULL) point " 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 "choicu: " parint "1. Create, 2,3. Insert, 4,5 Delte, 6,5each,7 display Case 1 : INPUT num-rodes, head = create Chead, num-Case 2: INPUT val, head sinsort-head (head, val).
case 3: INPUT index, val, head = insort-pos (head, val, index) case 4: head = del head Case 51 INPUT index, head=del-pos (head, index) case 6: INPUT val, search (head, val) case 7: display (heard)

Program code:

```
#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++)</pre>
        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)
```

```
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;
        1++;
    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 successfully...\n");
   else if(index==1)
       while(temp->next!=NULL)
            temp=temp->next;
        newnode->next=temp;
        temp->prev->next=newnode;
        newnode->prev=temp->prev;
        temp->prev=newnode;
        printf("\nNode inserted successfully...\n");
    }
   else
        while(temp->next!=NULL)
            if(i==index-1)
                break;
            temp=temp->next;
        newnode->next=temp->next;
        newnode->prev=temp;
        temp->next->prev=newnode;
        temp->next=newnode;
        printf("\nNode inserted successfully...\n");
```

```
}
    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=0,1=0;
    if(head==NULL)
        printf("\nList is empty...\n");
    else if(index==0)
        head=del_head(head);
        while(check->next!=NULL)
            check=check->next;
            1++;
        if(index>1)
            printf("\nInvalid index...\n");
        else if(index==1)
            while(temp1->next!=NULL)
                temp1=temp1->next;
            temp1->prev->next=NULL;
```

```
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);
        printf("\n%d found %d times\n",val,count);
void display(struct node *head)
    struct node *temp=head;
```

```
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: ");
```

```
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;
            display(head);
            break;
        case 8:
            exit(0);
        default:
            printf("\nInvalid choice...\n");
} while (choice!=8);
```

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
.Display
8.Exit
Enter your choice: 7
Displaying values...
Memory=00C80E00 Prev=00000000
                               Data=40 Next=00C80DE8
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: 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 head
5.Delete at the position
6.Search
7.Display
8.Exit
```

```
Enter your choice: 7
Displaying values...
                                    Data=40 Next=00C80E18
Memory=00C80E00 Prev=00000000
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...
```

```
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 head3.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
```

```
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 position
6.Search
7.Display
8.Exit
Enter your choice: 8
PS C:\Rohith\Backup\Desktop\SEM 3\Data Structures in C\Assignment-3>
```

Application

<u>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.</u>

Eg: Input: List2\$3

Output:List3

Pseudocode:

Application:	
Implement undo operation using doubly linked	
list. Input sequence of characters and	
create ande des avecus charactes. When 's'	and
create node for every character. When 's' is given as input delete the pravious	Reta
character.	Las
2g. input. Lista \$3 Output: List 3.	
Algorithm: - (wan = kand) 41	In
struct rode land the state of	Out
? char data:	sta
struct rode * rext, * prev;	if
Struct rode *rext, *parev;	
(ast-insert()	end
Input: head, char val	3
Output: head.	
newnode = (stauct rode * malloc(size of (stauct node))	end
if Chead == NUCL)	else
newnode > Daer = NULL	
newnoders next = NULL	
head=newnode	(10 the
end if	
else if Ctemp > next==NULL)	
newnode -> poer = temp newnode -> next = NULL	
temp -> next = newnode	end
end alse if	net
end else if else while (temp -> next 1= NULL)	
Cerry = 10-121-121-121	
end while	

	newhode > poer-temp () was as a
hed	par year
	newnode -> next = NOLL
n'\$'	arm eye
ory	return head.
	(ast-deletec)
	Input: staud rade *temp1, *temp2
	Output: staruet node * head
	struct node & temps, *temps
	if chead == Now) point "List Is empty"
	point "List Is empty"
Boo wheed	also it Ctempt - next = = Now.
400	End ele if www 200 200 200 200 200 200 200 200 200
	head = NULL
trode))	end ele if www 1 200 2 200 2 200 3 runs 300 2 200
x rugary)	else
	else temp1=head while (temp1 -> next!=NULL)
M	temp2=temp1
1 4	tempi= tempi-rext end while
q	force (tempi)
7	Land to the state of the state
3	end else
	neturn head

Display(). Input: staruet rode a head Output: void struct node * temp= head if (head== NULL)

point " 2 mpty List " () of
end if med a land a land else print tempodata temp= temp > next and else Main () Variables: struct node *head= NULL char input [= 0] INPUT input for i=o to stalen (input)

if (input [i] == '\$')

head = last_delete(head) end il else head = last_insert Chead, input cit) end for display (head)

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;
    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)
```

```
free(head);
        head=NULL;
    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++)</pre>
    {
        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.