EXP NO: 01

DATE:

IMPLEMENTATION OF SINGLY LINKED LIST AND ITS OPERATION

AIM: To write a c program to implement programs based on singly linked list.

PROGRAM1: ADD AN ELEMENT IN KTH POSITION

ALGORITHM:

STEP 1: Start

STEP 2: Create An object and allocate memory usinf the malloc function.

STEP 3: Get the value to be stored in the node ; store it in the data field. using the push function

Push the values in the linked list

STEP 4: Assign newnode next pointer to null.

STEP 5: Check the existence of the list .if(head==null) there is no list, the created new node is the first node of the list. Make the head pointer to point the headnode

STEP 6: If (head!=null)then there exist a list. Get the value and position of the list after which it has to be stored. Finally print the linked list.

STEP 7: Stop.

PROGRAM 1 PROCEDURE:

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
struct node{
        int value;
        struct node *next;
}*head=NULL;
void insert()
```

```
{
        int data;
        struct node*newnode;
        newnode=(struct node*)malloc(sizeof(struct node));
        printf("enter the data:");
        scanf("%d",&data);
        newnode->value=data;
        newnode->next=NULL;
        if(head==NULL)
        {
        head=newnode;
        }
        else
        {
           struct node*p;
        p=head;
        while(p->next!=NULL)
                p=p->next;
        newnode->next=head;
        head=newnode;
        }
}
void insert_at_k()
{
        int data,k,count=1;
        struct node*newnode,*p;
        newnode=(struct node*)malloc(sizeof(struct node));
        printf("enter the data:");
        scanf("%d",&data);
        printf("Enter the position to enter:");
        scanf("%d",&k);
        newnode->value=data;
        newnode->next=NULL;
        p=head;
        while(count!=k-1)
        {
                count++;
                p=p->next;
        }
        newnode->next=p->next;
        p->next=newnode;
}
void display()
{
                    struct node*p;
                    p=head;
                    printf("\n LIST:");
```

```
                    while(p!=NULL)
                    {
                            printf("->%d",p->value);
                            p=p->next;
                     }
}
int main()
{
                        int ch;
                        while(1)
                        {     printf("\n 1.insert\t 2.to insert at kth position \t3.Display  ");
                              printf("\nEnter ur choice :");
                              scanf("%d",&ch);
                           switch(ch)
                           {
                                case 1:
                                     insert();
                                     break;
                                  case 2:
                                    insert_at_k();
                                    break;
                                case 3:
                                        display();
                                           break;
                                case 4:
                                       exit(0);
                                              }
                        }
                        return 0;
}
```

PROGRAM 1 OUTPUT:

```
 1.insert        2.to insert at kth position    3.Display
Enter ur choice :1
enter the data:10

 1.insert        2.to insert at kth position    3.Display
Enter ur choice :1
enter the data:20

 1.insert        2.to insert at kth position    3.Display
Enter ur choice :1
enter the data:30

 1.insert        2.to insert at kth position    3.Display
Enter ur choice :3

 LIST:->30->20->10
 1.insert        2.to insert at kth position    3.Display
Enter ur choice :2
enter the data:25
Enter the position to enter:2

 1.insert        2.to insert at kth position    3.Display
Enter ur choice :3

 LIST:->30->25->20->10
 1.insert        2.to insert at kth position    3.Display
Enter ur choice :4

--------------------------------
Process exited after 25.58 seconds with return value 0
Press any key to continue . . .
```

PROGRAM 2: REVERSE A LINKED LIST

ALGORITHM:

STEP 1: Start

STEP 2: Create a structure using the struct Node i.e.,int data,stuct Node*next.

STEP 3: Get the value to be stored in the node ; store it in the data field. using the push function

Push the values in the linked list

STEP 4: To reverse a linked list create a user defined function called interReverseLL()

4.1: Initialize Node* current = head,Node *prev = NULL, *after = NULL.

4.2:using while loop check if the condition (current != NULL) and do the following

after = current->next;

current->next = prev;

prev = current;

current = after;

4.3: finally after the loop ends make head = prev.

STEP 5:using print function display the linked list then display the reversed linked list.

STEP 6: Stop.

PROGRAM 2 PROCEDURE:

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
struct node{
        int value;
        struct node *next;
}*head=NULL;
void insert()
{
        int data;
        struct node *newnode,*p;
        newnode=(struct node*)malloc(sizeof(struct node));
        printf("\nEnter data:");
        scanf("%d",&data);
        newnode->value=data;
        newnode->next=NULL;
        if(head==NULL)
        head=newnode;
        else
        {
                p=head;
                while(p->next!=NULL)
```

```
                p=p->next;
                p->next=newnode;
        }
}
void display(struct node *h)
{
        struct node *p;
        p=h;
        printf("\n");
        while(p->next!=NULL)
        {
                printf("%d->",p->value);
                p=p->next;
        }
        printf("%d->",p->value);
        printf("NULL");
}
void reverse(struct Node* head_ref)
{
   struct Node* prev = NULL;
   struct Node* current = *head_ref;
   struct Node* next = NULL;
   while (current != NULL) {
       next = current->next;
      current->next = prev;
       prev = current;
       current = next;
   }
   *head_ref = prev;
}
int main()
{
```

```
                int ch;
                while(1)
                {      printf("\n 1.insert\t 2.reverse \t3.display\t4.exit  ");
                       printf("\nEnter choice :");
                       scanf("%d",&ch);
                   switch(ch)
                   {
                        case 1:
                             insert();
                             break;
                         case 2:
                           reverse(head);
                           break;
                        case 3:
                                display(head);
                                  break;
                        case 4:
                               exit(0);
                                      }
                }
                return 0;
}
```

PROGRAM 2 OUTPUT:

```
1.insert        2.reverse       3.display       4.exit
Enter choice:1
Enter data:10
1.insert        2.reverse       3.display       4.exit
Enter choice:1
Enter data:20
1.insert        2.reverse       3.display       4.exit
Enter choice:1
Enter data:30
1.insert        2.reverse       3.display       4.exit
Enter choice:1
Enter data:40
1.insert        2.reverse       3.display       4.exit
Enter choice:3
10->20->30->40->NULL
1.insert        2.reverse       3.display       4.exit
Enter choice:2
1.insert        2.reverse       3.display       4.exit
Enter choice:3
40->30->20->10->NULL
1.insert        2.reverse       3.display       4.exit
Enter choice:4
--------------------------------
Process exited after 0.1409 seconds with return value 15
Press any key to continue . . .
```

PROGRAM 3: REVERSE A LINKED LIST IN K-GROUP.

ALGORITHM:

STEP 1: start
STEP 2: Create the list by following steps 3-7
STEP 3: Create the object and allocates memory for object using malloc function
STEP 4: Get the value to be stored in the node store it in the data field and assign new node next pointer to NULL.
STEP 5: Check the existence of list by checking the head pointer is equivalent to null or not
STEP 6: If head==NULL then there is no list already existing creating new node is the first node of the list by making the head pointer to point to the new node.
STEP 7: If head!=NULL then there exist a list already
7.1: To find the last node assign the head value to pointer p and traverse the list until p->next becomes NULL and assign the newnode to p->next
STEP 8: To reverse the list in kth group follow step 8.1-10
8.1: First find the number of element in the list using pointer p
8.2: Intialize i=1 and increment i until p->next becomes null\
STEP 9: After finding the total number of elements(i) get k from user
9.1: create pointer p points to head and moves n/k times
9.2: create another pointer p1 which points to pointer p move the pointer p1 for k times
9.3: swap the value of p and p1 using temporary variables

STEP 10: Display the list
STEP 11: Stop

PROGRAM 3 PROCEDURE:

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

struct node
{
 int value;
 struct node *next;
}*head=NULL,*head1=NULL;

void insert()
{
 int data;
 struct node *newnode;
 newnode=(struct node*)malloc (sizeof(struct node));
 printf("enter the value:");
 scanf("%d",&data);
 newnode->value=data;
 newnode->next=NULL;
 if(head==NULL)
 {
  head=newnode;
 }
 else
 {
  struct node *p;
  p=head;
  while(p->next != NULL)
  {
   p=p->next;
  }
  p->next=newnode;
        }
}
void display()
{
 struct node *p;
 p=head;
 while(p->next != NULL)
 {
  printf("%d-> ",p->value);
  p=p->next;
 }
```

```
 printf("%d ",p->value);
}
int count()
{
 struct node *p;
 p=head;
 int i=1;
 while(p->next != NULL)
 {
  i++;
  p=p->next;
 }
 return i;
}
void reverse(struct node *p,int i)
{
 struct node*p1;
 int temp;
 int j,k;
 (i%2==0)?(k=i/2):(k=i/2+1);

          while(k--)
 {
  j=1;
  p1=p;
  while(j<i)
  {
   p1=p1->next;
   j++;
  }

  temp=p->value;
  p->value=p1->value;
  p1->value=temp;

  p=p->next;
  i=i-2;
 }}
void reversekth()
{
 struct node *p;
 int k,i,j;
 printf("enter the value of k:");
 scanf("%d",&k);
 p=head;
 int n=count();
 for(i=0;i<n/k;i++)
```

```
 {
  reverse(p,k);
  j=0;
  while(j<k)
  {
   p=p->next;
   j++;
  }
 }
}

int main()
{
 int choice;
 while(1)
 {
            printf("\n1.create a list\t");
 printf("2.reverse in the list\t");
 printf("3.display\t");
 printf("4.exit");
  printf("\nenter the choice:");
  scanf("%d",&choice);
  switch(choice)
  {
   case 1:
    insert();
    break;
   case 2:

    reversekth();
    printf("\nlist after the reverse operation:");
    display();
    break;
   case 3:
           printf("\nlist:");
    display();
    break;
   case 4:
    exit(0);
  }
 }
 return 0;
}
```

PROGRAM 3 OUTPUT:

```
1.create a list 2.reverse in the list   3.display       4.exit
enter the choice:1
enter the value:10

1.create a list 2.reverse in the list   3.display       4.exit
enter the choice:1
enter the value:20

1.create a list 2.reverse in the list   3.display       4.exit
enter the choice:1
enter the value:30

1.create a list 2.reverse in the list   3.display       4.exit
enter the choice:1
enter the value:40

1.create a list 2.reverse in the list   3.display       4.exit
enter the choice:1
enter the value:50

1.create a list 2.reverse in the list   3.display       4.exit
enter the choice:1
enter the value:60

1.create a list 2.reverse in the list   3.display       4.exit
enter the choice:1
enter the value:70

1.create a list 2.reverse in the list   3.display       4.exit
enter the choice:1
enter the value:80

1.create a list 2.reverse in the list   3.display       4.exit
enter the choice:2
enter the value of k:3

list after the reverse operation:30-> 20-> 10-> 60-> 50-> 40-> 70-> 80
1.create a list 2.reverse in the list   3.display       4.exit
enter the choice:4

--------------------------------
Process exited after 36.01 seconds with return value 0
Press any key to continue . . .
```

DESCRIPTION	MAXIMUM MARK	MARKS SCORED
OBSERVATION	30	
RECORD	20	
TOTAL	50	

RESULT:

Thus the all the three given programs based on singly linked list are executed and outputs are verified.

EXP NO: 02

DATE:

IMPLEMENTATION OF DOUBLY LINKED LIST AND ITS OPERATION

AIM: To write a c program to implement programs based on doubly linked list.

PROGRAM 1: APPEND THE DOUBLY LINKED LISTS

ALGORITHM:

STEP 1: Start

STEP 2: Create a structure using the struct Node i.e.,int data,stuct Node*next, stuct Node*prev.

STEP 3: Get the value to be stored in the node ; store it in the data field. using the INSERT function insert the values in the linked list

STEP 4: To append an element in a linked list create a user defined function called void insertLL()

4.1: Initialize *head=NULL;

STEP 5: Assign newnode next and prev point to null

STEP 6 : check the existence of list using by checking the head point is equal to null or not

STEP 7 : If **(head==NULL)** then there is no list already existing the created newnode is the newnode of the list

7.1: make the head pointer to point to newnode

STEP 8 : If the head pointer is not equal to null then there exist a list

8.1:To find the last node assign the head value in pointer initialized called p

8.2: Traverse the list until p next becomes NULL

8.3: Assign address of newnode to p next and address of p to newnode prev

STEP 9: Then create a function to append the list and display it

STEP 10: In the function pass the parameters struct node *he1,struc node *he2

STEP 11: Assign the he1 to temporary pointer p1 and he2 to temporary pointer p2

STEP 12: Traverse the p1 next until NULL and then assign h2 to p1 next and p1 to p2 prev

STEP 13: Now free the h2 by free() function

STEP 14: Stop.

PROGRAM 1 PROCEDURE:

```
#include<stdio.h>
#include<stdlib.h>
#include<conio.h>
struct node{
        int value;
        struct node *next,*prev;
}*h1=NULL,*h2=NULL;
void insert(struct node *h)
{
        int data;
        struct node *newnode,*p;
        newnode=(struct node *)malloc(sizeof(struct node));
        printf("Enter value:");
        scanf("%d",&data);
        newnode->next=NULL;
        newnode->prev=NULL;
        newnode->value=data;
        p=h;
        if(h==NULL)
        h1=newnode;
        else{
                while(p->next!=NULL)
                {
                        p=p->next;
                }
                newnode->prev=p;
                p->next=newnode;
                newnode->next=NULL;
        }
```

```
}
void append(struct node *he1,struct node *he2)
{
        struct node *p1,*p2;
        p1=he1;
        p2=he2;
        while(p1->next!=NULL)
        {
                p1=p1->next;
        }
        p2->prev=p1;
        p1->next=p2;
        free(he2);
}
void display(struct node *he)
{
        struct node *p;
        p=he;
        while(p->next!=NULL)
        {
                printf("%d-->",p->value);
                p=p->next;
        }
printf("%d-->NULL",p->value);
}
int main()
{
        int choice;
        while(1)
        {
                printf("\n1.insert first\n2.insert second\n3.append");
```

```
            printf("\nEnter choice");
            scanf("%d",&choice);
            switch(choice)
            {
                case 1:
                    insert(h1);
                    display(h1);
                    break;
                case 2:
                    insert(h2);
                    display(h2);
                    break;
                case 3:
                    append(h1,h2);
                    display(h1);
                    break;
                default:
                    exit(0);

            }
        }
}
```

PROGRAM 1-OUTPUT:

```
1.insert first
2.insert second
3.append
Enter choice1
Enter value:10
10-->NULL
1.insert first
2.insert second
3.append
Enter choice1
Enter value:20
10-->20-->NULL
1.insert first
2.insert second
3.append
Enter choice2
Enter value:30
30-->NULL
1.insert first
2.insert second
3.append
Enter choice2
Enter value:40
30-->40-->NULL
1.insert first
2.insert second
3.append
Enter choice3
10-->20-->30-->40-->NULL
1.insert first
2.insert second
3.append
Enter choice8

--------------------------------
Process exited after 21.82 seconds with return value 0
Press any key to continue . . .
```

PROGRAM 2: REORDER A DOUBLY **LINKED LIST**

ALGORITHM:

STEP 1: start
STEP 2: Create the list by following steps 3-7
STEP 3: Create the object and allocates memory for object using malloc function
STEP 4: Get the value to be stored in the node store it in the data field and assign new node next pointer to NULL.
STEP 5: Check the existence of list by checking the head pointer is equivalent to null or not
STEP 6: If head==NULL then there is no list already existing creating new node is the first node of the list by making the head pointer to point to the new node.

STEP 7: If head!=NULL then there exist a list already
7.1: To find the last node assign the head value to pointer p and traverse the list until p->next becomes NULL and assign the newnode to p->next
STEP 8: To reorder the list follow step 8-10
STEP 9: Assign the prev as head and curr as head->next
STEP 10: Traverse curr till end
10.1: if curr->data < prev->data then swap the curr and prev
10.2: if curr->next and cur->next->data > curr->data then swap curr and curr->next
10.3: assign prev as curr->next and curr as curr->next->next
STEP 11: Stop

PROGRAM 2 PROCEDURE:

```
#include <stdio.h>

#include <stdlib.h>

typedef struct Node {


   int data;


   struct Node* next;

} Node,*head=NULL;

void rearrange(Node* head)

{


   if (head == NULL)

      return;


   Node *prev = head, *curr = head->next;

   while (curr) {


      if (prev->data > curr->data) {

         int temp = prev->data;

         prev->data = curr->data;

         curr->data = temp;

      }

      if (curr->next && curr->next->data > curr->data)
```

```
                {
        int temp = curr->next->data;
        curr->next->data = curr->data;
        curr->data = temp;
      }
      prev = curr->next;
      if (!curr->next)
        break;
      curr = curr->next->next;
   }
}

void insert()
{
        int data;
        struct node*newnode;
        newnode=(struct node*)malloc(sizeof(struct node));
        printf("enter the data:");
        scanf("%d",&data);
        newnode->value=data;
        newnode->next=NULL;
        if(head==NULL)
        {
        head=newnode;
        }
        else
        {
           struct node*p;
        p=head;
        while(p->next!=NULL)
                p=p->next;
```

```
        newnode->next=head;
        head=newnode;
        }
}

void display(Node* head)
{
   Node* curr = head;
   while (curr->next != NULL) {
      printf("%d->", curr->data);
      curr = curr->next;
   }
   printf("%d",curr->data);
}

int main()
{
                        int ch;
                        while(1)
                        {     printf("\n 1.insert\t 2.reorder the list\t3.Display  ");
                              printf("\nEnter ur choice :");
                              scanf("%d",&ch);
                           switch(ch)
                           {
                                case 1:
                                     insert();
                                     break;
                                  case 2:
                                     rearrange(head);
                                     break;
                                case 3:
```

```
                display(head);
                    break;
            case 4:
                exit(0);
                    }
        }
        return 0;
}
```

PROGRAM 2 OUTPUT:

```
 1.insert        2.reorder the list     3.Display
Enter ur choice :1
enter the data:10
 1.insert        2.reorder the list     3.Display
Enter ur choice :1
enter the data:20
 1.insert        2.reorder the list     3.Display
Enter ur choice :1
enter the data:30
 1.insert        2.reorder the list     3.Display
Enter ur choice :1
enter the data:40
 1.insert        2.reorder the list     3.Display
Enter ur choice :1
enter the data:50
 1.insert        2.reorder the list     3.Display
Enter ur choice :1
enter the data:60
 1.insert        2.reorder the list     3.Display
Enter ur choice :3
10->20->30->40->50->60->NULL
Enter ur choice :2

 1.insert        2.reorder the list     3.Display
Enter ur choice :3
10->60->20->50->30->40->NULL
Enter ur choice :4
--------------------------------
Process exited after 0.1673 seconds with return value 19
Press any key to continue . . .
```

PROGRAM 3: REVERSE A DOUBLY LINKED LIST IN K GROUP .

ALGORITHM:

STEP 1: start

STEP 2: Create the list by following steps 3-7
STEP 3: Create the object and allocates memory for object using malloc function
STEP 4: Get the value to be stored in the node store it in the data field and assign new node next pointer to head.
STEP 5: Check the existence of list by checking the head pointer is equivalent to null or not
STEP 6: If head==NULL then there is no list already existing creating new node is the first node of the list by making the head pointer to point to the new node.
STEP 7: If head!=NULL then there exist a list already
7.1: To find the last node assign the head value to pointer p and traverse the list until p->next becomes NULL and assign the newnode to p->next
STEP 8: To reverse the list in kth group follow step 8.1-10
8.1: First find the number of element in the list using pointer p
8.2: Intialize i=1 and increment i until p->next becomes null\
STEP 9: After finding the total number of elements(i) get k from user
9.1: create pointer p points to head and moves n/k times
9.2: create another pointer p1 which points to pointer p move the pointer p1 for k times
9.3: swap the value of p and p1 using temporary variables
STEP 10: Display the list
STEP 11: Stop

PROGRAM 3 PROCEDURE:

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


struct node
{
   struct node *prev;
   int value;
   struct node *next;
}*head=NULL,*head1=NULL;


void insert()
{
 int data;
 struct node *newnode;
 newnode=(struct node*)malloc (sizeof(struct node));
 printf("enter the value:");
 scanf("%d",&data);
 newnode->prev=NULL;
 newnode->value=data;
```

```
 newnode->next=NULL;
 if(head==NULL)
 {
  head=newnode;
 }
 else
 {
  struct node *p;
  p=head;
  while(p->next != NULL)
  {
   p=p->next;
  }
  p->next=newnode;
  newnode->prev=p;
        }
}
void display()
{
 struct node *p;
 p=head;
 while(p->next != NULL)
 {
  printf("%d ",p->value);
  p=p->next;
 }
 printf("%d ",p->value);
}
int count()
{
 struct node *p;
 p=head;
 int i=1;
 while(p->next != NULL)
```

```
 {
  i++;
  p=p->next;
 }
 return i;
}
void reverse(struct node *p,int i)
{
 struct node*p1;
 int temp;
 int j,k;
 (i%2==0)?(k=i/2):(k=i/2+1);

          while(k--)
 {
  j=1;
  p1=p;
  while(j<i)
  {
   p1=p1->next;
   j++;
  }

  temp=p->value;
  p->value=p1->value;
  p1->value=temp;

  p=p->next;
  i=i-2;
 }}
void reversekth()
{
 struct node *p;
 int k,i,j;
```

```
 printf("enter the value of k:");
 scanf("%d",&k);
 p=head;
 int n=count();
 for(i=0;i<n/k;i++)
 {
  reverse(p,k);
  j=0;
  while(j<k)
  {
   p=p->next;
   j++;
  } }}
int main()
{
 int choice;
 while(1)
 {
          printf("\n1.create a list\t");
 printf("2.reverse in the list\t");
 printf("3.display\t");
 printf("4.exit");
  printf("\nenter the choice:");
  scanf("%d",&choice);
  switch(choice)
  {
   case 1:
    insert();
    break;
   case 2:

    reversekth();
    printf("\nlist after the reverse operation:");
    display();
```

```
   break;
  case 3:
         printf("\nlist:");
   display();
   break;
  case 4:
   exit(0);
 }
}
return 0;
}
```

PROGRAM 3 OUTPUT:

```
1.create a list 2.reverse in the list   3.display       4.exit
enter the choice:1
enter the value:1

1.create a list 2.reverse in the list   3.display       4.exit
enter the choice:1
enter the value:2

1.create a list 2.reverse in the list   3.display       4.exit
enter the choice:1
enter the value:3

1.create a list 2.reverse in the list   3.display       4.exit
enter the choice:1
enter the value:4

1.create a list 2.reverse in the list   3.display       4.exit
enter the choice:1
enter the value:5

1.create a list 2.reverse in the list   3.display       4.exit
enter the choice:1
enter the value:6

1.create a list 2.reverse in the list   3.display       4.exit
enter the choice:1
enter the value:7

1.create a list 2.reverse in the list   3.display       4.exit
enter the choice:2
enter the value of k:2

list after the reverse operation:2-> 1-> 4-> 3-> 6-> 5-> 7
1.create a list 2.reverse in the list   3.display       4.exit
enter the choice:4

--------------------------------
Process exited after 38.88 seconds with return value 0
Press any key to continue . . .
```

DESCRIPTION	MAXIMUM MARK	MARKS SCORED
OBSERVATION	30	
RECORD	20	
TOTAL	50	

RESULT:

Thus the all the three given programs based on singly linked list are executed and outputs are verified.

EXP NO: 03

DATE:

APPLICATION OF LINKED LIST

AIM:

To implement a c program for the application of the linked list .

PROGRAM 1: REMOVE THE OCCURANCE IN THE LIST.

ALGORITHM:

STEP 1: start.
STEP 2: allocate memory for the nodes
STEP 3: get the nodes of the list.
STEP 4: assign the p to the head.
STEP 5: while(p->next!=head)
5.1:check if(p->value==p->next->value) if the condition is true go to step 5.2 else step 5.3
5.2:then initialize q=p->next->next and increment p->next=q;
5.3:increment p as p=p->next;
STEP 6: print the list.
STEP 7:stop

PROGRAM 1-CODING:

```
#include<stdio.h>
#include<conio.h>
#include<malloc.h>
struct node
{
    int value;
    struct node*next;
    }*head=NULL;
    void insertlast()
    {
        int data;
        struct node*newnode;
        newnode=(struct node*)malloc(sizeof(struct node));
        printf("enter the data:");
        scanf("%d",&data);
        newnode->value=data;
        newnode->next=NULL;
        if(head==NULL)
        {
```

```
head=newnode;
newnode->next=head;
}
else
{
   struct node*p;
p=head;
while(p->next!=head)
p=p->next;
newnode->next=p->next;
p->next=newnode;
}
}
void occurance()
{
    struct node*p,*q;
    p=head;
    while(p->next!=head)
    {
                if(p->value==p->next->value)
                {
                                q=p->next->next;
                                p->next=q;
                                }
                                p=p->next;
                                }
                                }
        void display()
        {
            struct node*p;
            p=head;
            while(p->next!=head)
            {
                        printf("%d->",p->value);
                        p=p->next;
                        }
                        printf("%d",p->value);
                        }

          int main()
          {    printf("\n1-insert a node\n2-remove occurance\n3-display");
              int ch;
              while(1)
              {
                    printf("\n enter your choice");
                    scanf("%d",&ch);
                    switch(ch)
```

```
                    {
                        case 1:
                            insertlast();
                            break;
                            case 2:
                                printf("\nafter the removal of occurance:");
                                occurance();
                                display();
                                break;
                                case 3:
                                    printf("\n LIST:");
                                display();
                                break;
                                case 4:
                                    exit(0);
                                    }
                                    }
                                    }
```

PROGRAM 1-OUTPUT:

```
1-insert a node
2-remove occurance
3-display
 enter your choice1
enter the data:10

 enter your choice1
enter the data:20

 enter your choice1
enter the data:30

 enter your choice1
enter the data:40

 enter your choice1
enter the data:40

 enter your choice3

 LIST:10->20->30->40->40
 enter your choice2

after the removal of occurance:10->20->30->40
 enter your choice4

--------------------------------
Process exited after 42.12 seconds with return value 0
Press any key to continue . . .
```

PROGRAM 2: POLYNOMIAL MULTIPLICATION

ALGORITHM:
STEP 1: Start
STEP 2: Create object and allocate memory using malloc function.
STEP 3: Get the value to be stored in the node ; store it in the data field
STEP 4: assign null to next field of newnode.
STEP 5: insert datas to the list and second list
STEP 6: multiply the coefficient of first node of first list to all the coefficients of second node
STEP 7: add the powers
STEP7.1: continue step 6 and 7 till multiplying all the coefficients of first node with second list
STEP 8: add the coefficients of same powers
STEP 9: print the result list
STEP 8: Stop

PROGRAM 2-CODING:

```
#include<stdio.h>
#include<conio.h>
#include<malloc.h>
struct node {
int cof;
int exp;
struct node *next;
}*head1=NULL,*head2=NULL,*head4=NULL;
void create(struct node **head) {
struct node *newnode,*p;
newnode=(struct node*)malloc(sizeof(struct node));
int co,ex;
printf("enter coefficient:");
scanf("%d",&co);
printf("enter expononent:");
scanf("%d",&ex);
newnode->cof=co;
newnode->exp=ex;
newnode->next=NULL;
if(*head==NULL)
{
*head=newnode;
}
else if((*head)->exp<newnode->exp) {
newnode->next=*head;
*head=newnode;
}
else{
p=*head;
while(p->next!=NULL && p->next->exp > newnode->exp) {

p=p->next;
```

```
}
newnode->next=p->next;
p->next=newnode;
}
}
void insert(struct node *newnode) {
struct node *p;
p=head4;
if(head4==NULL)
{
head4=newnode;
}
else if((newnode)->exp>head4->exp) {
newnode->next=head4;
head4=newnode;
}
else{
p=head4;
while(p->next!=NULL && p->next->exp > newnode->exp) {
p=p->next;
}
newnode->next=p->next;
p->next=newnode;
}
}
void add()
{
struct node *p,*temp;
p=head4;
while(p->next!=NULL)

{
if(p->exp==p->next->exp)
{
p->cof+=p->next->cof;
temp=p->next;
p->next=temp->next;
free(temp);
}
else
{
p=p->next;
}
}
}
void create1() {
int i=1,ch;
while(i==1) {
create(&head1);
printf("do you want to continue:\n (0),(1)");
scanf("%d",&ch);
if(ch==0) {
i=0;
```

```
}
}
}
void create2() {
int i=1,ch;
while(i==1) {
create(&head2);
printf("do you want to continue?(no=0)(yes=1)\n");

scanf("%d",&ch);
if(ch==0) {
i=0;
}
}
}
void display(struct node**head) {
struct node *p;
p=*head;
while(p!=NULL) {
printf("%dx^%d ",p->cof,p->exp);
p=p->next;
}
}
void display1() {
display(&head1);
}
void display4() {
display(&head4);
}
void display2() {
display(&head2);
}
void multiply()
{
struct node *newnode,*p1,*p2;
p1=head1;
while(p1!=NULL)
{
p2=head2;
while(p2!=NULL)

{
newnode=(struct node*)malloc(sizeof(struct node));
newnode->cof=p1->cof*p2->cof;
newnode->exp=p1->exp+p2->exp;
newnode->next=NULL;
insert(newnode);
p2=p2->next;
}
p1=p1->next;
}
add();
display4();
```

```
}
int main() {
int i=1,ch;
printf("1.create 1st list\n");
printf("2.create 2nd list\n");
printf("3.display 1st list\n");
printf("4.display 2nd list\n");
printf("5.multiply of two polynomial\n");
printf("6.exit");
while(i==1) {
printf("\n\nenter your choice:");
scanf("%d",&ch);
switch(ch) {
case 1:
create1();
break;
case 2:
create2();
break;

case 3:
display1();
break;
case 4:
display2();
break;
case 5:
multiply ();
break;
case 6:
i=0;
break;
}
}
return 0;

}
```

PROGRAM 2-OUTPUT:

```
1.create 1st list
2.create 2nd list
3.display 1st list
4.display 2nd list
5.multiply of two polynomial
6.exit

enter your choice:1
enter coefficient:2
enter expononent:2
do you want to continue:
 (0),(1)1
enter coefficient:3
enter expononent:1
do you want to continue:
 (0),(1)0


enter your choice:2
enter coefficient:3
enter expononent:2
do you want to continue?(no=0)(yes=1)
1
enter coefficient:2
enter expononent:0
do you want to continue?(no=0)(yes=1)
0


enter your choice:5
6x^4 9x^3 4x^2 6x^1

enter your choice:6

--------------------------------
Process exited after 102.2 seconds with return value 0
Press any key to continue . . .
```

DESCRIPTION	MAXIMUM MARK	MARKS SCORED
OBSERVATION	20	
RECORD	05	
TOTAL	25	

RESULT:

Thus the all the two given programs based on application of linked list are executed and outputs are verified.

EXP NO: 04

DATE:

IMPLEMENTATION OF STACK ADT

AIM: To write a c program to implement programs based on linked list.

PROGRAM 1: IMPLEMENTATION OF STACK USING ARRAY AND LINKED LIST

ALGORITHM:

STEP 1: Start

STEP 2: Create a structure using the struct Node i.e.,int data,stuct Node*next for implementing using linked list and int stack[5] and top = -1 for array implementation.

STEP 3: Get the value to be stored in the node ; store it in the data field. using the push function.Push the values in the linked list similarly for array get the value from the user and push by stack[++top]=data.

STEP 4: To pop the elements using array use top--.And to pop using linked list make struct node*temp if(head==NULL) then print the stack is empty else make temp=head, head=head->next and free(temp).

STEP 5:to display elements using array int i,for(i=top;i>=0;i--) print stack[i].for linked list initialize struct node*p , p=head.while(p->next!=NULL) print p->value then increament by p=p->next finally print p->value

STEP 6: Stop.

PROGRAM 1-CODING:

Using array:

```
#include<stdio.h>
int stack[5];
int top=-1;
void push()
{
   int data;
   printf("enter the value : ");
   scanf("\n %d",&data);
   stack[++top]=data;
}
```

```
void pop()
{
    top--;
}
void display()
{
    int i;
    for(i=top;i>=0;i--)
    {
       printf("%d ",stack[i]);
    }
}
int main()
{
   int choice=0;
   printf("\n1.Push the element\n2.Pop the element\n3.display\n4.End");
   while(1)
   {
     printf("\nenter your choice : ");
     scanf("%d",&choice);
     switch(choice)
   {
      case 1:
           push();
           break;
      case 2:
           pop();
           break;
      case 3:
           display();
           break;
```

```
      default:
           exit(1);
           break;
    }
}
}
```

Using linked list:

```
#include<stdio.h>
#include<malloc.h>
struct node
{
      int value;
      struct node*next;
}*head=NULL;
void push()
{
    int data;
    printf("\nenter the values : ");
    scanf(" \n %d",&data);
    struct node*newnode;
    newnode=(struct node*)malloc(sizeof(struct node));
    newnode->value=data;
    newnode->next=NULL;
    if(head==NULL)
    {
       head=newnode;
       }
    else
    {
        newnode->next=head;
        head=newnode;
```

```
    }
    }

    void pop()
    {
        struct node*temp;
        if(head==NULL)
        {
          printf("the stack is empty");
        }
        else
        {
           temp=head;
           head=head->next;
           free(temp);
         }
         }

   void display()
   {
       struct node*p;
       p=head;
       while(p->next!=NULL)
       {
         printf("\n %d",p->value);
         p=p->next;
       }
     printf("\n %d",p->value);
   }

int main()
```

```
{
   int choice=0;
   printf("\n1.Push the element\n2.Pop the element\n3.display\n4.End");
   while(1)
   {
    printf("\nenter your choice : ");
    scanf("%d",&choice);
    switch(choice)
   {
     case 1:
          push();
          break;
     case 2:
          pop();
          break;
     case 3:
          display();
          break;
     default:
          exit(1);
          break;
   }
}
}
```

PROGRAM 1-OUTPUT:

Using array:

```
1.Push the element
2.Pop the element
3.display
4.End
enter your choice : 1
enter the value : 10

enter your choice : 1
enter the value : 20

enter your choice : 1
enter the value : 30

enter your choice : 2

enter your choice : 3
20 10
enter your choice : 1
enter the value : 40

enter your choice : 3
40 20 10
enter your choice : 4

--------------------------------
Process exited after 20.1 seconds with return value 1
Press any key to continue . . .
```

Using linked list:

```
1.Push the element
2.Pop the element
3.display
4.End
enter your choice : 1
enter the value : 5

enter your choice : 1
enter the value : 10

enter your choice : 1
enter the value : 15

enter your choice : 3
15 10 5
enter your choice : 2

enter your choice : 3
10 5
enter your choice : 4

--------------------------------
Process exited after 57 seconds with return value 1
Press any key to continue . . .
```

PROGRAM 2 EVALUATE POSTFIX EXPRESSION USING STACK

ALGORITHM:

STEP 1: Start

STEP 2: Create a structure using the struct Node i.e.,int data,stuct Node*next.

STEP 3: Get the value to be stored in the node ; store it in the data field. using the push function.Push the values in the linked list

STEP 4: first sort the array using the sort stack function.

4.1: if (!isEmpty(*s)) then make int x = pop(s),sortStack(s)then call the function

sortedInsert(s, x);

4.2: if (isEmpty(*s) || x > top(*s)) then do push(s, x),int temp = pop(s);

sortedInsert(s, x),push(s, temp);

STEP 5:using print function display the linked list then display the rotated linked list.

STEP 6: Stop.

PROGRAM 2-PROCEDURE:

```
#include <stdio.h>
#include <conio.h>
#include <ctype.h>
#include <string.h>
#define MAX 100
char st[MAX];
int top=–1;
void push(char st[], char);
char pop(char st[]);
Stacks 235
void InfixtoPostfix(char source[], char target[]);
int getPriority(char);
int main()
{
char infix[100], postfix[100];
clrscr();
printf("\n Enter any infix expression : ");
```

```
gets(infix);
strcpy(postfix, "");
InfixtoPostfix(infix, postfix);
printf("\n The corresponding postfix expression is : ");
puts(postfix);
getch();
return 0;
}
void InfixtoPostfix(char source[], char target[])
{
int i=0, j=0;
char temp;
strcpy(target, "");
while(source[i]!='\0')
{
 if(source[i]=='(')
 {
 push(st, source[i]);
 i++;
 }
 else if(source[i] == ')')
 {
 while((top!=–1) && (st[top]!='('))
 {
target[j] = pop(st);
j++;
 }
 if(top==–1)
 {
 printf("\n INCORRECT EXPRESSION");
exit(1);
```

```
 }
 temp = pop(st);//remove left parenthesis
 i++;
 }
 else if(isdigit(source[i]) || isalpha(source[i]))
 {
target[j] = source[i];
j++;
 i++;
 }
 else if (source[i] == '+' || source[i] == '–' || source[i] == '*' ||
source[i] == '/' || source[i] == '%')
 {
 while( (top!=–1) && (st[top]!= '(') && (getPriority(st[top])
> getPriority(source[i])))
 {
target[j] = pop(st);
j++;
 }
 push(st, source[i]);
 i++;
 }
 else
236 Data Structures Using C
 {
 printf("\n INCORRECT ELEMENT IN EXPRESSION");
 exit(1);
 }
}
while((top!=–1) && (st[top]!='('))
{
```

```
target[j] = pop(st);
j++;
}
target[j]='\0';
}
int getPriority(char op)
{
if(op=='/' || op == '*' || op=='%')
 return 1;
else if(op=='+' || op=='–')
return 0;
}
void push(char st[], char val)
{
if(top==MAX–1)
 printf("\n STACK OVERFLOW");
else
{
 top++;
 st[top]=val;
}
}
char pop(char st[])
{
char val=' ';
if(top==–1)
 printf("\n STACK UNDERFLOW");
else
{
 val=st[top];
 top—;
```

```
}

return val;

}
```

PROGRAM 2-OUTPUT:

```
enter a postfix expression: 3 5 2 * +
evaluated value = 13
--------------------------------
Process exited after 0.1342 seconds with return value 58
Press any key to continue . . .
```

DESCRIPTION	MAXIMUM MARK	MARKS SCORED
OBSERVATION	20	
RECORD	05	

TOTAL	**25**	

RESULT:

Thus the all the given programs based on linked list are executed and outputs are verified.