12. Linked List

Aim

Create a linked list

Algorithm

I struct Node

1. START
2. int data
3. struct Node *link
4. STOP

II Create struct pointers as start and temp and set start to NULL value.

III void insertfront(int item)

1. START
2. create struct Node pointer as p and allocate its size
3. p→data = item
4. if (start == NULL)
   1. p→link = NULL;
   2. start = p

5. Else

1. p → link = start
2. start = p;

6 End If

7 STOP

IV void insertend (int item)

1. START
2. struct Node *p; and allocate space
3. p → data = item;
4. p → link = NULL;
5. if (start = NULL) {
   1. start = p;
6. Else
   1. temp = start
   2. while (temp → link != NULL)
      1. temp = temp → link
   3. End while.
   4. temp → link = p
7. End if
8. STOP

V void insertany(int item, int pos)

1. START
2. Create a struct pointer(p) and allocate space
3. p → data = item
4. p → link = NULL
5. if (start == NULL)
    1. start = p
6. else if (pos <= 1)
    1. p → link = start
    2. start = p;
7. else
    1. temp = start
    2. for (int i=1 to pos-1)
        1. temp = temp → link
    3. p → link = temp → link
    4. temp → link = p
8. STOP

V void delfront()

1. START
2. if (start == NULL)
   1. print("List Empty")
3. Else
   1. if (start → link == NULL) then
      1. temp = start
      2. start = NULL
   2. Else
      1. temp = start
      2. start = start → link
   3. End If
4. End If
5. print " temp → data + "removed"
6. free(temp);
7. STOP

VI void delend()

1. START
2. check start == NULL and start → link = NULL as above
3. Else
   1. temp = start

2. while (temp → link → link != NULL)

temp = temp → link

3. End While

4. print (" temp → link → data + " is removed ")

5. free (temp → link)

6. temp → link = NULL

7. STOP

VIII void delany (int pos)

1. START
2. check &start = NULL like above
3. else if (pos < 1)
   1. check like delfront and deallocate the pointer
4. Else if (pos == size - 1)
   1. check like delend and deallocate the node
5. Else
   1. iterate till the position
   2. print the value and deallocate
6. STOP.

IX Void display()

1. if (start == NULL)
   1. print "Empty list"
2. Else
   1. temp = start
   2. while (temp != NULL)
      1. print (temp->data + " -> ")
      2. temp = temp -> link.
   3. End while
3. End if.
4. STOP

X void main()

1. START
2. Show menu by printing on console
3. Input the choice of user
4. And Input item and pos wherever necessary
5. STOP

Output

Obtained & Verified

```
//SLL

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

static int size=0;

struct Node {
        int data;
        struct Node *link;
} *start=NULL ,*temp ;


void insertfront(int item){
        struct Node *p;
        p= (struct Node*)malloc(sizeof(struct Node));
        p->data=item;
        if(start==NULL){
                p->link=NULL;
                start=p;
        }
        else{
                p->link=start;
                start=p;
        }
        size++;
}
void insertend(int item){
        struct Node *p;
        p= (struct Node*)malloc(sizeof(struct Node));
        p->data=item;
        p->link=NULL;
        if(start==NULL){
                start=p;
        }
        else{
                temp=start;
                while(temp->link!=NULL){
                        temp=temp->link;
                }
                temp->link=p;
        }
        size++;
}
void insertany(int item,int pos){
        struct Node *p;
        p= (struct Node*)malloc(sizeof(struct Node));
        p->data=item;
        p->link=NULL;
```

```
        if(start==NULL){
                start=p;
        }
        else if(pos<=1){
            p->link=start;
            start=p;
        }
        else{
                temp=start;
                for(int i=1 ;i<pos-1 ;i++)
                        temp=temp->link;

                p->link=temp->link;
                temp->link=p;
        }
        size++;
}
void deletefront(){
        if(start==NULL)
                printf("List Empty\n");
        else{
            if(size==1){
          temp=start;
          start=NULL;
            }
            else{
                temp=start;
                start=start->link;
            }

                printf("%d is removed from front\n",temp->data);
                free(temp);
                size--;
        }
}
void deletend(){
        if(start==NULL)
                printf("List Empty\n");
        else{
            if(size==1){
          printf("%d is removed from end\n",start->data);
          temp=start;
          start=NULL;
          free(temp);
            }
            else{
                temp=start;
                struct Node *loc;
                loc=temp->link;
                while(loc->link!=NULL)
```

```
                        loc=loc->link;
                        temp=temp->link;
                printf("%d is removed from end\n",loc->data);
                temp->link=NULL;
                free(loc);
                size--;
            }
        }
}
void deleteany(int pos){
        if(start==NULL || size==0)
                printf("List Empty\n");
    else if (pos<1){
        printf("%d is removed from front\n",start->data);
        if(size==1){
          temp=start;
          free(temp);
          start=NULL;
        }
        else{
          temp=start;
          start=start->link;
          free(temp);
        }
    }
    else if(pos==size-1){

            if(size==1){
          temp=start;
          start=NULL;
          printf("%d is removed from end\n",temp->data);
          free(temp);
            }
            else{
                temp=start;
                struct Node *loc;
                loc=temp->link;
                while(loc->link!=NULL)
                        loc=loc->link;
                        temp=temp->link;
                printf("%d is removed from end\n",loc->data);
                temp->link=NULL;
          free(loc);
            }
    }
        else{
            if(size==1){
          temp=start;
          start=NULL;
          printf("%d is removed from front\n",temp->data);
```

```
          free(temp);
            }
            else{
                temp=start;
                struct Node *loc;
                loc=temp->link;
                for(int i=1; i<pos-1; i++){
                        loc=loc->link;
                        temp=temp->link;
                }
                printf("%d is removed \n",loc->data);
                temp->link=loc->link;

                free(loc);
            }
        }
        size--;
}
void display(){
        if(start==NULL)
                printf("List Empty\n");
        else{
                printf("");
                temp=start;
                while(temp->link!=NULL){
                        printf("%d ->",temp->data);
                        temp=temp->link;
                }
                printf("%d\n",temp->data);
        }
}

int main(){
        int choice,item,pos;
        printf("1...display\n");
  printf("2...insertfront\n");
        printf("3...insertend\n");
        printf("4...insertany\n");
  printf("5...deletefront\n");
        printf("6...deletend\n");
        printf("7...deleteany\n");
  printf("8...quit\n");
        printf("else...menu\n");

    int quit=1;
    while(quit!=0){
        printf("\nOption : ");
        scanf("%d",&choice);
        switch(choice){
                case 1: display();
```

```
                    break;
                case 2: printf("item: ");
                    scanf("%d",&item);
                        insertfront(item);
                        break;
                case 3: printf("item: ");
                    scanf("%d",&item);
                        insertend(item);
                        break;
                case 4: printf("item: ");
                    scanf("%d",&item);
                    printf("position: ");
                        scanf("%d",&pos);
                        if(pos<=size)
                            insertany(item,pos);
                        else
                            printf("total size: %d\t [%d : %d)\n",size,0,size);
                        break;
                case 5: deletefront();
                        break;
                case 6: deletend();
                        break;
                case 7: printf("position: ");
                    scanf("%d",&pos);
                    if(pos<=size-1 || size==0)
                            deleteany(pos);
                        else
                            printf("total size: %d\t [%d : %d]\n",size,0,size-1);
                        break;
                case 8: quit=0;
                    printf("*****Program aborted*****");
                    break;
                default:
                                printf("\n1...display\n");
                            printf("2...insertfront\n");
                                printf("3...insertend\n");
                                printf("4...insertany\n");
                            printf("5...deletefront\n");
                                printf("6...deletend\n");
                                printf("7...deleteany\n");
                            printf("8...quit\n");
                            printf("else...menu\n");

        }
    }

    return 0;

}
```

```
1...display
2...insertfront
3...insertend
4...insertany
5...deletefront
6...deletend
7...deleteany
8...quit
else...menu

Option : 2
item: 1

Option : 2
item: 2

Option : 2
item: 3

Option : 3
item: 4

Option : 1
3 ->2 ->1 ->4

Option : 4
item: 5
position: 3

Option : 1
3 ->2 ->5 ->1 ->4

Option : 5
3 is removed from front
```

```
Option : 5
2 is removed from front

Option : 6
4 is removed from end

Option : 7
position: 1
5 is removed from end

Option : 6
1 is removed from end

Option : 6
List Empty

Option : 1
List Empty

Option : 9

1...display
2...insertfront
3...insertend
4...insertany
5...deletefront
6...deletend
7...deleteany
8...quit
else...menu

Option : 8
*****Program aborted*****
```