13 - Doubly Linked List

Aim

Create a doubly linked list

Algorithm

I create a node of struct data type having data, *next, *prev inside

II create start, tail and temp pointer also.

III void insertfront(int item)

1. START
2. Create a struct pointer p and allocate some space
3. p → data = item
4. if (start == NULL)
   1. p → next = NULL
   2. start = p
   3. tail = p
   4. p → prev = NULL
5. Else

1. start → prev = p
2. p → next = start
3. start = p
4. start → prev = NULL

6. End If

76. STOP

IV void insertend (int item)

1. START
2. create struct ptr *p and allocate space
3. p→data = item
4. if (start == NULL)
   1. same as ~~above~~ insertfront()
5. else {
   1. tail → next = p
   2. tail → next → prev = tail
   3. p → next = NULL
   4. tail = p
6. End If
7. STOP

V void insertany (int item, int pos)

1. START
2. create a struct ptr *p and allocate it.
3. same as ~~3 5 front~~ steps 3 to 5 from insert front ()
4. Else if (pos == 0)
   1 same as steps 5.1 to 5.4 from insertfront()
5. Else if (pos == size)
   1 same as steps 5.1 to 5.4 from insertend()
6. Else
   1. temp = start
   2. for (int i = 1 to pos-2)
      1. temp = temp → next
   3. p → next = temp → next
   4. temp → next → prev = p
   5. temp → next = p
   6. p → prev = temp
7. End If
8. STOP

III void delfront()

1. START
2. check if list is empty and print empty if it is
3. else
   1 print (start → data + " is removed")
   2 if (start → next == NULL)
      1. temp = start
      2. free (temp)
      3. start = tail = NULL
   3. else
      1. temp = start
      2. start = start → next
      3. start → prev = NULL
      3. free (temp)
   4. End If
4. End If
5. STOP

IV void delend()

1. START
2. repeat steps 2 to 3. 2-3
3. else
   3 else

1. tail = tail → prev
2. temp = tail → next
3. tail → next = NULL
4. free (temp)

5. End If

4 End If

5. STOP

V void delany(int pos)

1. START
2. if (pos < 1)
   1. Repeat steps 3.1 to 3.4 in delfront()
3. else if (pos == size-1)
   1. Repeat steps 3.3.1 to 3.3.4 in delend()
4. else
   1. temp = start
   2. struct Node *loc;
   3. for(int i to pos-2)
      temp = temp → next
   4. loc = temp → next
   5. temp → next = temp → next → next
   6. temp → next → prev = temp

7 free (loc)

5& End If

6. STOP

VI void display ()

1 START
2 if (start == NULL)
    1 print ("List Empty").
3 else
    1 temp = start
    2 while (temp != NULL)
        1. print (temp->data + " --> ");
    3. End while
4. End if
5 STOP

VII int main ()

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

Output

Obtained & Verified.

```
//DLL

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

static int size=0;

struct Node{
  int data;
  struct Node *next;
  struct Node *prev;
} *start=NULL ,*tail=NULL ,*temp;


void insertfront(int item){
  struct Node *p;
  p= (struct Node*)malloc(sizeof(struct Node));
  p->data=item;
  if(start==NULL){
    p->next=NULL;
    start=p;
    tail=p;
  }
  else{
    start->prev=p;
    p->next=start;
    start=p;
  }
  start->prev=NULL;
  size++;
}
void insertend(int item){
  struct Node *p;
  p= (struct Node*)malloc(sizeof(struct Node));
  p->data=item;
  p->next=NULL;
  if(start==NULL){
    start=p;
    tail=p;
    start->prev=NULL;
  }
  else{

    tail->next=p;
    tail->next->prev=tail;
    p->next=NULL;
    tail=p;
  }
  size++;
}
```

```
void insertany(int item,int pos){
  struct Node *p;
  p= (struct Node*)malloc(sizeof(struct Node));
  p->data=item;
  p->next=NULL;
  if(start==NULL){
    start=p;
        tail=p;
    start->prev=NULL;
  }
  else if(pos==0){
    start->prev=p;
    p->next=start;
    start=p;
  }
  else if(pos==size){
    tail->next=p;
    p->prev=tail;
    p->next=NULL;
    tail=p;
  }else if(pos==size-1){
      p->prev=tail->prev;
      p->prev->next=p;
      tail->prev=p;
      p->next=tail;
  }
  else{
    temp=start;
    for(int i=1 ;i<pos-1 ;i++)
      temp=temp->next;

    p->next=temp->next;
    temp->next->prev=p;

    temp->next=p;
    p->prev=temp;
  }
  size++;
}
void deletefront(){
  if(start==NULL)
    printf("List Empty\n");
  else{
    printf("%d is removed from front\n",start->data);
    if(size==1){
      temp=start;
      free(temp);
      start=NULL;
      tail=NULL;
    }
```

```
    else{
      temp=start;
      start=start->next;
      start->prev=NULL;
      free(temp);
    }
    size--;
  }
}
void deletend(){
  if(start==NULL)
    printf("List Empty\n");
  else{
    printf("%d is removed from end\n",tail->data);
        if(size==1){
          temp=start;
          free(temp);
          start=NULL;
          tail=NULL;
        }
        else{
        tail=tail->prev;
        temp=tail->next;
        tail->next=NULL;
        }
    free(temp);
    size--;
  }
}
void deleteany(int pos){
  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->next;
      start->prev=NULL;
      free(temp);
    }
  }
  else if(pos==size-1){
    printf("%d is removed from end\n",tail->data);
    tail=tail->prev;
    temp=tail->next;
    tail->next=NULL;
```

```
    free(temp);
  }
  else{
    temp=start;
    struct Node *loc;
    for(int i=1 ;i<pos-2 ;i++){
      temp=temp->next;
    }
    loc=temp->next;
    temp->next=temp->next->next;
    temp->next->prev=temp;

    printf("%d is removed from pos[%d]\n",loc->data,pos);

    free(loc);
  }
  size--;

}
void display(){
  if(start==NULL)
    printf("List Empty\n");
  else{
    printf("\n\t___foreward list___\n");
    temp=start;
    while(temp->next!=NULL){
      printf("%d ->",temp->data);
      temp=temp->next;
    }
    printf("%d\n",temp->data);
  }
}
void revdisp(){
  if(start==NULL)
    printf("List Empty\n");
  else{
    printf("\n\t___reverse list___\n");
    temp=tail;
    while(temp->prev!=NULL){
      printf("%d ->",temp->data);
      temp=temp->prev;
    }
    printf("%d\n",temp->data);
  }
}

int main(){
  int choice,item,pos;
  printf("-1...reverse display\n");
  printf("1...forward 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: revdisp();
        break;
      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 is %d\t [0-%d]\n",size,size);
        break;


      case 5: deletefront();
        break;


      case 6: deletend();
        break;
```

```
        case 7: printf("position: ");
            scanf("%d",&pos);
            if(pos<=size-1)
              deleteany(pos);
            else{
              if(size>0)
                printf("Total size: %d\t [0-%d]\n",size,size-1);
              else
                printf("Empty list\n");
            }
            break;


        case 8: quit=0;
            printf("*****Program aborted*****\n\n");
            break;


        default:
        printf("\n-1...reverse display\n");
        printf("1...forward 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...reverse display
1...forward 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

     ___foreward list___
3 ->2 ->1 ->4

Option : -1

     ___reverse list___
4 ->1 ->2 ->3

Option : 4
item: 5
```

```
Option : 4
item: 5
position: 2

Option : 1

     ___foreward list___
3 ->5 ->2 ->1 ->4

Option : 5
3 is removed from front

Option : 5
5 is removed from front

Option : 6
4 is removed from end

Option : 7
position: 1
1 is removed from end

Option : 7
position: 0
2 is removed from front

Option : 1
List Empty

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