18/12/20 Lab 9 IBM19CS006

Doubly linked list

```
#include <stdio.h>
#include <stdlib.h>
struct node{
 int data;
 struct node *next;
 struct node * prev;
};
struct node *head=NULL;
void create() {
 struct node * newnode, *temp;
 int item;
 newnode =(struct node*)malloc(sizeof(struct node));
 printf("Enter the data");
 scanf("%d",&item);
 newnode -> ~~data~~ data=item;
 newnode -> next=NULL;
 newnode -> prev=NULL;
 if(head==NULL)
 {  head=newnode;
    }
 else {
 temp=head;
 while (temp->next!=NULL)
 {  temp=temp->next;
   }

 temp -> next= newnode;
 newnode -> next=NULL;
 newnode-> prev=temp;
}
}
```

Scanned with CamScanner

1BM19CS006

```
void insert_beg(){
 struct node* newnode;
  newnode = (struct node*) malloc(sizeof(struct node));
 printf("Enter the item\n");
 scanf("%d", &newnode->data);
  newnode->next=NULL;
  newnode->prev=NULL;
 if(head==NULL) {
   head=newnode;
  }
 else{
  newnode->next=head;
  head->prev=newnode;
   head=newnode;
  }
 }
void insert_end(){
 struct node *temp,*newnode;
 newnode=(struct node*)malloc(sizeof (struct node));
 printf("Enter the item\n");
 scanf("%d",&newnode->data);
  newnode->next =NULL;
  newnode->prev =NULL;
if (head ==NULL) {
  head=newnode;
 }
else{
  temp=head;
 while(temp->next!=NULL)
temp=temp→next;
 temp→next=newnode;
  newnode→prev=temp;
 }
}
void insert_after(){
int listele;
 struct node *newnode, *temp;
 printf("Enter element after which new element
  should be entered in the list \n");
  scanf("%d", &listele);
 newnode=(struct node*)malloc(sizeof(struct node));
 printf("Enter newnode data \n");
 scanf("%d", &newnode→data);
 newnode→next=NULL;
 newnode→prev=NULL;
if(head==NULL){
  printf("Empty list \n");
    return;
  }
temp=head;
while(temp→data!=listele){
 temp=temp→next;
  if(temp==NULL)
  {   printf("Element not found \n");
       return;
    }
  }
```

```
newnode -> next = temp -> next;
temp -> next = newnode;
newnode -> prev = temp;
newnode -> next -> prev = newnode;
}

void insert_before(){
int listele;
struct node *newnode, *temp;
printf("Enter element before which new element
   should be entered in list \n");
 scanf("%d", &listele);
 newnode = (struct node*)malloc(sizeof(structnode));
 printf("Enter the newnode data \n");
 scanf("%d", &newnode->data);
newnode -> next = NULL;
newnode -> prev = NULL;
if(head == NULL){
   printf("Empty list \n");
     return;
  }

 temp = head;
 while(temp -> data != listele){
     temp = temp -> next;
    if(temp == NULL)
  {    printf("Element not found \n");
          return;
    }
  }
newnode -> prev = temp -> prev;
```

```
temp->prev=newnode;
newnode->next=temp;
newnode->prev->next=newnode;
}
void del(){
struct node*temp;
int ele;
if(head==NULL){
  printf("Empty list\n");
   return;
}
printf("Enter the element to be deleted\n");
scanf("%d",&ele);
temp=head;
while(temp->data!=ele)
{  temp=temp->next;
   if(temp==NULL)
  {    printf("Elements not found\n");
       ~~br~~ return;
     }
   }
 if(temp==head){
    head=head->next;
   }
 else if(temp->next==NULL)
  {  temp=temp->prev;
     temp->next=NULL;
    }
```

```
else{
  temp->prev->next = temp->next;
  temp->next->prev = temp->prev;
 }
}

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

int main() {
   int choice;
   do {
   printf ("1.Create \n 2.Inser at start \n 3.Insert at end
    \n 4.Insert after a node \n 5.Insert before a node
 \n 6.Delete \n 7.Display \n 8.Exit \n");
   printf ("Enter your choice \n");
   scanf ("%d", &choice);
   switch (choice) {
   case 1: create(); break;
   case 2: insert_beg(); break;
   case 3: insert_end(); break;
   case 4: insert_after(); break;
   case 5: insert_before(); break;
   case 6: del(); break;
   ~~case~~
```

1BM19CS006

```
case 7: display(); break;
case 8: exit(0);
  }
 }while(choice!=8);
return 0;
}
```

Scanned with CamScanner