Maintenance Of Linked List

Code Id	25
Date Updated	3/7/2010
Title	Maintenance of linked list

Description

This program is an implementation of linked list where items can be placed anywhe

Codes Snippet

```
#include
#include
#define NULL ??

struct node
{
        int info;
        struct node *next;
};
struct node *start=NULL;

/*NODE CREATION*/
/*function to allocate memory for a new node*/
struct node*create_node(int data)
{
        struct node *nw;
        nw = (struct node *)malloc(sizeof(struct node));
        nw -> info = data;
        nw -> next = NULL;
        return nw;
}

/* INSERTION */
/*function to add node in the beginning of the linked list*/
void add_beg(struct node *nw)
{
        if (start == NULL)
                start=nw;
        else
{
                nw->next=start;
                start=nw;
        }
}
/*function to add node in the end of the linked list*/
void add_end(struct node *nw)
{
        struct node *ptr;
        if (start == NULL)
                start=nw;
        else
{
        for( ptr=start; ptr->next != NULL; ptr=ptr->next);
ptr->next=nw;
        }
}

/*function to add node at a particular position of the linked list*/
void add_pos(struct node *nw, int pos)
{
        int choice,position;
struct node *ptr;
        if (start == NULL)
        {
                printf(?Linked list is empty, would you like to add the node ?(1/
                scanf(?%d?,&choice);
                if (choice==1)
                        start=nw;
        }
        else
```

```
                        scanf(“%d”,&choice);
                        if (choice==1)
                                add_end(nw);
                }
        }
}

/* DELETION*/
/*function to delete node from the beginning of the linked list*/
void del_beg()
{
        if (start == NULL)
                printf(“Sorry, can’t delete. The list is empty”);
        else
                start=start->next;
}
/*function to delete node from the end of the linked list*/
void del_end()
{
        struct node *ptr;
        if (start == NULL)
                printf(“Sorry, can’t delete. The list is empty”);
        else
{
        for( ptr=start; (ptr->next)->next != NULL; ptr=ptr->next);
ptr->next= NULL;
        }
}
/*function to delete node from a particular position of the linked list*/
void add_pos(int pos)
{
        int choice,position;
struct node *ptr;
        if (start == NULL)
                printf(“Sorry, can’t delete. The list is empty”);
        else
{
        for( ptr=start, position=1; position < pos && ptr ; position ++,ptr=ptr->
        if (ptr != NULL)
                ptr -> next = (ptr -> next) -> next;
        else
        {
                        printf(“nThe position entered is outside the list”);
printf(“nWould you like to delete the last node ?(1/0)”);
                        scanf(“%d”,&choice);
                        if (choice==1)
                                delete_end();
                }
        }
}

/*TRAVERSAL */
void traverse( )
{
        struct node *ptr;
        if (start == NULL)
                printf(“EMPTY LIST”);
        else
{
        for( ptr=start; ptr; ptr=ptr->next)
printf(“%d t”,ptr->info);
        }
}

/*MODIFY*/
/*function to modify the information in a node at a particular position of the li
void modify(int pos, int nwinfo)
{
        int position;
struct node *ptr;
        if (start == NULL)
                printf(“Linked list is empty);
        else
{
        for( ptr=start, position=0; position < pos && ptr ; position ++,ptr=ptr->
        if (ptr != NULL)
                ptr -> info = nwinfo;
        else
                        printf(“nThe position entered is outside the list”);
```

```
                printf(“n2. MODIFY”);
                printf(“n3. DELETE”);
                printf(“n4. TRAVERSE”);
                printf(“n5. EXIT”);
                printf(“nn Enter your choice”);
                scanf(“%d”,&choice);
                switch(choice)
                {
                        case 1:
                                printf(“nEnter information for the new node”);
                                scanf(“%d”,&data);
                                nw=(struct node *)malloc(sizeof(struct node));
                                nw=create_node(nw);
                                printf(“n1. INSERT NODE IN THE  BEGINNING”);
                                printf(“n2. INSERT NODE AT THE  END”);
                                printf(“n3. INSERT NODE AT A PARTICULAR POSITION”
                                printf(“n4. RETURN”);
                        printf(“nn Enter your choice”);
                                scanf(“%d”,&choice1);
                switch(choice)
                        {
                                        case 1:
                                                add_beg(nw);
                                                break;
                                        case 2:
                                                add_end(nw);
                                                break;
                                        case 3:
                                                printf(“nEnter the position”);
                                                scanf(“%d”,&position);
                                                add_pos(nw,position);
                                                break;
                                        case 4:
                                                break;
                                }
                        }while(choice1!=4);
                        case 2:
printf(“nEnter the position of the node you want to change”);
scanf(“%d”,&position);
printf(“nEnter the new information”);
scanf(“%d”,&nwdata);
modify(position,nwdata);
break;
                        case 3:
printf(“n1. DELETE NODE FROM THE  BEGINNING”);
                                printf(“n2. DELETE NODE FROM THE  END”);
                                printf(“n3. DELETE NODE AT A PARTICULAR POSITION”
                                printf(“n4. RETURN”);
                        printf(“nn Enter your choice”);
                                scanf(“%d”,&choice1);
                switch(choice)
                        {
                                        case 1:
                                                del_beg();
                                                break;
                                        case 2:
                                                del_end();
                                                break;
                                        case 3:
                                                printf(“nEnter the position”);
                                                scanf(“%d”,&position);
                                                del_pos(position);
                                                break;
                                        case 4:
                                                break;
                                }
                        }while(ch!=4);
                        case 4:
                                traverse();
                        case 5:
                                exit(0);
                }
        }while(choice!=5);
        return 0;
}
```