JOEL NINAN JOHNSON
1BM19CS199

14/12/2020

LAB-8 : Double linked List Implementation

Q. WAP Implement doubly linked list with primitive operations
a) Create a doubly linked list
b) Insert a new node to the left of the node
c) Delete the node based on a specific value
d) Display the contents of the list.

A.
```
struct node
{
    struct node *prev;
    int data;
    struct node *next;
}


void delete(){
    struct node *ptr, struct node *temp;
    int val;
    printf("Enter the value:");
     scanf("%d", &val);
     temp=head;
     while(temp->data!=val)
         temp=temp->next;
     if (temp->next==NULL)
         printf(" Cannot be deleted");
    else if(temp->next->next==NULL)
         temp->next=NULL;
         printf(" Last Node Deleted);
    else
    {
         ptr=temp->next;
         temp->next=ptr->next;
```

JOEL NINAN JOHNSON
1BM19CS199

classmate
Date
Page

14/12/2020

```
        ptr -> next -> prev = temp;
        free (ptr);
        printf(" Node Deleted");
    }
}

void display (){
    struct node *ptr = head;
    ~~while(ptr->next != NULL)~~
    if ( ptr == head)
        printf(" Empty List");
    else
    {
        printf(" \n\n LIST -> ");
        while (ptr != NULL)
        {
            printf(" \t %d", ptr->data);
            ptr = ptr->next;
        }
    }
}

void Create_List(){
    struct node *ptr;
    int i, n, new_data;
    print (" Enter the no. of nodes");
    scanf( "%d", &n);

    if (n>=1)
    {
        head = (struct node *) malloc (sizeof(struct node));
        if (head != NULL)
```

JOEL NINAN JOHNSON
1BM19CS199

14/12/2020

```
{
    printf("Enter value for Node 1 :");
    scanf("%d", &new_data);

    head -> data = new_data;
    head -> prev = NULL;
    head -> next = NULL;
    ~~head data~~ last = head;
    for (i=2; i<=n; i++)
    {
        ptr=(struct node *)malloc(sizeof(struct node
        if (ptr!=NULL)
        {
            printf("Enter value for Node %d:", i)
            scanf("%d", &new_data);
            ptr -> data = new_data;
            ptr -> prev = last;
            ptr -> next = NULL;

            last -> next = ptr;
            last = ptr;
        }
    } printf("~~~~ Linked List Created");
    }
}
else
{
    printf(" ~~a~~ Nodes cannot be created ~~selected~~ ");
}
}
}
```

JOEL NINAN JOHNSON
1BM19CS199

classmate
Date
Page

14/12/2020

```
void insert() {
        int i; position, new_data;
        struct node *ptr, *temp;

        if (head == NULL)
        {
                printf(" List is Empty ");
        }
        else
        {
                temp=head; i=1;
                while (i<position -1 && temp!=NULL)
                {
                        temp= temp->next;
                        i++;
                }
                if (position == 1)
                {
                        ptr ->data= ~~data~~ new_data;
                        ptr -> next = head;
                        ptr -> prev = NULL;
                        head -> prev = ptr;
                        head = ptr;
                }
                else if (temp==last)
                {
                        ptr->data=new_data;
                        ptr -> next=NULL;
                        ptr ->prev= last;
                        last ->next = ptr;
                        last = ptr;
                }
```

JOEL NINAN JOHNSON
1BM19CS199

14/12/2020

```
                else if(temp!=NULL)
                {
                    ptr->data= new_data;
                     ptr->next = temp->next;
                      ptr->prev= temp;
                     if(temp->next!=NULL)
                     {
                           temp->next->prev= ptr;
                      }
                      temp->next= ptr;
                 }
                 else
                 { printf("Invalid position");
                  }
          }
    }
```