01/12/2015

7).
a) WAP to 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

Pseudo code:

```
Void createList(int n){
Struct node {
    int data;
    struct node *prev,*next;
};
Void createList(int n) {
    for(i=1;i<=n;i++){
        Print" enter data for node %d",i);
        Scanf("%d",&data);
    }newnode=(struct node*)malloc(sizeof(struct node));
    newnode->data=data;
    newnode->prev=newnode->next=NULL;
    If (head==NULL)
        head=tail=newnode;
    }else{
      tail->next=newnode;
      newnode->prev=tail;
       tail=newnode
    }
}
}
```

insertion

```
void insertAtFront(int data){
    newnode->data=data;
    newnode->prev=newnode->next=NULL;
    newnode->next=head;
    if(head==NULL){
        head=tail=newnode;
    }else{
        head->prev=newnode;
        head=newnode;
    }}

void insertAtEnd(int data){
    newnode->next=NULL;
    newnode->prev=tail;
    if(tail==NULL){
        head=tail=newnode;
    }else{
        tail->next=newnode;
        tail=newnode;
    }}

void deletebyvalue(int value){
    if(head==NULL){print "List is Empty"}
    else
    while(temp!=NULL && temp->data!=value){
        temp=temp->next
    }
    if(temp==NULL){print "value not found"}
    if(temp==head){
        "Empty cond"
        temp=head
        head=head->next
        if(head!=NULL){head->prev=NULL
}
```

```
else if (temp == tail){
        deleteatEnd();

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

    free(temp);
}
}
```

Execute

code:-

```
#include <stdio.h>
#include <stdlib.h>
struct node{
    int data;
    struct node *next, *prev;
};
struct node *head=NULL;
struct node *tail=NULL;
void createlist(int n){
    struct node *newnode;
    int i, data;
    for(i=1; i<=n; i++){
    printf("\nEnter the data for node %d: ", i);
    scanf("%d", &data);
    newnode = (struct node*)malloc(sizeof(struct node));
    newnode->data = data;
    newnode->prev = newnode->next = NULL;
    if(head==NULL){
        head = tail = newnode;
    } else{
        tail->next = newnode;
        newnode->prev = tail;
        tail = newnode;
}}}
```

```
void InsertAtLeft(int data){
    struct node *newnode = (struct node*)malloc
        (sizeof(struct node));
    newnode->data = data;
    newnode->prev = NULL;
    newnode->next = head;
    if(head==NULL){
        head = tail = newnode;
    }else{
        head->prev = newnode;
        head = newnode;
    }
}
void InsertAtLast(int data){
    struct node *newnode = (struct node*)malloc
        (sizeof(struct node));
    newnode->data = data;
    newnode->prev = tail;
    newnode->next = NULL;
    if(tail==NULL){
        head = tail = newnode;
    }else{
        tail->next = newnode;
        tail = newnode;
    }
}
void DeleteByValue(int value){
    struct node *temp = head;
    if(head==NULL){
        printf("The List is Empty!\n");
        return;
    }
    while(temp!=NULL && temp->data!=value){
        temp = temp->next;
    }
    if(temp==NULL){
        Printf("\n Value not found!");
```

```
        return;
    }
    if(temp==head){
        struct node *temp;
        if(head==NULL){
            printf("List is empty.\n");
        return;
    }
    temp=head;
    head=head->next
    if(head!=NULL){
        head->prev=NULL;
    }else{
        tail=NULL;
    }
    free(temp);
}else if(temp==tail){
    struct node*temp;
    if(tail==NULL){
        printf("List is empty.\n");
        return;
    }
    temp=tail;
    tail=tail->prev;
    if(tail!=NULL){
        tail->next=NULL;
    }else{
        head=NULL;
        free(temp);
    }
}else{
    temp->next->prev=temp->prev;
    temp->prev->next=temp->next;
    free(temp);
}
```

```
void display(){
    struct node *temp=head;
    if(head==NULL){
        printf("List is empty.");
    }
    else{
        Printf("Linked List:\n");
        while(temp!=NULL){
            Printf("%d ",temp->data);
            temp=temp->next;
        }
    }
}

void main(){
    int ch,value,n,data;
    do{
    Printf("\n1) Create linked list 2) Insert at
Left 3) Insert at end. 4) Delete by value 5) Display
list");
    Printf("\nEnter -1 to exit");
    Printf("Enter your chorce:");
    scanf("%d",&ch);
    Switch(ch){
        case 1:
            Printf("Enter no. of nodes:");
            Scanf("%d",&n);
            Create_list(n);
            break;
        case 2:
            printf("Enter the data to insert:");
            Scanf("%d",&data);
            InsertAtLeft(data);
            break;
        Case 3;
            Printf("\nEnter the data to insert:");
            Scanf("%d",&data);
```

```
            InsertAtLast(data);
            break;
        case 4:
            printf("Enter value:");
            scanf("%d",&value);
            DeleteByValue(value);
            break;
        Case 5:
            display();
            break;
        default:
            Printf("Invalid input.");
    }while(ch!=-1);
}
```

1) create linked list 2) Insert at left 3) Insert at end 4) Delete by value 5) display list

Enter -1 to exit
Enter your choice:1
Enter no. of nodes=5

Enter the data for node1:2
Enter data: 3
Enter data:4
Enter data: 5
Enter data: 6

1) Create linked list 2) Insert at left 3) Insert at end 4) Delete by value 5) display list
Enter -1 to exit
Enter your choice:2
Enter the Data to insert:1

1) Create linked list 2) Insert at left 3) Insert at end 4) Delete by value 5) display list
Enter -1 to exit
Enter your choice:5
Linked List:
1 2 3 4 5 6

1) Create linked list 2) Insert at left 3) Insert at end 4) Delete by value 5) display list
Enter -1 to exit
Enter your choice:3
Enter the data to insert:7

1) create linked list 2) Insert at left 3) Insert at end 4) Delete by value 5) Display list
Enter -1 to exit
Enter your choice:4
Enter value:5

1) Create linked list 2) Insert at left 3) Insert at end 4) Delete by value 5) Display list
Enter -1 to exit
Enter your choice:5
Linked list;
1 2 3 4 6 7

1) Create linked list 2) Insert at left 3) Insert at end 4) Delete by value 5) Display list
Enter -1 to exit
Enter your choice -1
Invalid Input.