Day 3

1. Write a function “insert\_any()” for inserting a node at any given position of the linked list. Assume

position starts at 0.

class Node {

int data;

Node nextNode;

}

class LinkedList {

private Node head;

// {

public void insert(int data) {

//create a new Node and store a data.

Node node = new Node();

node.data = data;

node.nextNode = null;

//check the head is null or not.

//if head is null, assign the Node and exit.

if (this.head == null) {

head = node;

return;

}

//assign a head into the temp Node and loop it until find the null reference.

Node tempNode = this.head;

while (tempNode.nextNode != null) {

tempNode = tempNode.nextNode;

}

//assign new node.

tempNode.nextNode = node;

}

// }

public void insertNth(int data, int position) {

//create new node.

Node node = new Node();

node.data = data;

node.nextNode = null;

if (this.head == null) {

//if head is null and position is zero then exit.

if (position != 0) {

return;

} else { //node set to the head.

this.head = node;

}

}

if (head != null && position == 0) {

node.nextNode = this.head;

this.head = node;

return;

}

Node current = this.head;

Node previous = null;

int i = 0;

while (i < position) {

previous = current;

current = current.nextNode;

if (current == null) {

break;

}

i++;

}

node.nextNode = current;

previous.nextNode = node;

}

// {

public void print() {

if (this.head == null) {

return;

}

//print all nodes

Node tempNode = this.head;

while (tempNode != null) {

System.out.print(tempNode.data + "->");

tempNode = tempNode.nextNode;

}

System.out.println("NULL");

}

// }

}

public class Main {

public static void main(String[] args) {

LinkedList ls = new LinkedList();

ls.insert(10);

ls.insert(20);

ls.insert(30);

ls.print();

ls.insertNth(25, 2);

ls.print();

}

}

1. Write a function “delete\_beg()” for deleting a node from the beginning of the linked list.
2. Write a function “delete\_end()” for deleting a node from the end of the linked list.

Sol. #include<stdio.h>

#include<conio.h>

#include<process.h>

struct node

{

int data;

struct node \*next;

}\*start=NULL,\*q,\*t;

int main()

{

int ch;

void insert\_beg();

void insert\_end();

int insert\_pos();

void display();

void delete\_beg();

void delete\_end();

int delete\_pos();

while(1)

{

printf("\n\n---- Singly Linked List(SLL) Menu ----");

printf("\n1.Insert\n2.Display\n3.Delete\n4.Exit\n\n");

printf("Enter your choice(1-4):");

scanf("%d",&ch);

switch(ch)

{

case 1:

printf("\n---- Insert Menu ----");

printf("\n1.Insert at beginning\n2.Insert at end\n3.Insert at specified position\n4.Exit");

printf("\n\nEnter your choice(1-4):");

scanf("%d",&ch);

switch(ch)

{

case 1: insert\_beg();

break;

case 2: insert\_end();

break;

case 3: insert\_pos();

break;

case 4: exit(0);

default: printf("Wrong Choice!!");

}

break;

case 2: display();

break;

case 3: printf("\n---- Delete Menu ----");

printf("\n1.Delete from beginning\n2.Delete from end\n3.Delete from specified position\n4.Exit");

printf("\n\nEnter your choice(1-4):");

scanf("%d",&ch);

switch(ch)

{

case 1: delete\_beg();

break;

case 2: delete\_end();

break;

case 3: delete\_pos();

break;

case 4: exit(0);

default: printf("Wrong Choice!!");

}

break;

case 4: exit(0);

default: printf("Wrong Choice!!");

}

}

return 0;

}

void insert\_beg()

{

int num;

t=(struct node\*)malloc(sizeof(struct node));

printf("Enter data:");

scanf("%d",&num);

t->data=num;

if(start==NULL) //If list is empty

{

t->next=NULL;

start=t;

}

else

{

t->next=start;

start=t;

}

}

void insert\_end()

{

int num;

t=(struct node\*)malloc(sizeof(struct node));

printf("Enter data:");

scanf("%d",&num);

t->data=num;

t->next=NULL;

if(start==NULL) //If list is empty

{

start=t;

}

else

{

q=start;

while(q->next!=NULL)

q=q->next;

q->next=t;

}

}

int insert\_pos()

{

int pos,i,num;

if(start==NULL)

{

printf("List is empty!!");

return 0;

}

t=(struct node\*)malloc(sizeof(struct node));

printf("Enter data:");

scanf("%d",&num);

printf("Enter position to insert:");

scanf("%d",&pos);

t->data=num;

q=start;

for(i=1;i<pos-1;i++)

{

if(q->next==NULL)

{

printf("There are less elements!!");

return 0;

}

q=q->next;

}

t->next=q->next;

q->next=t;

return 0;

}

void display()

{

if(start==NULL)

{

printf("List is empty!!");

}

else

{

q=start;

printf("The linked list is:\n");

while(q!=NULL)

{

printf("%d->",q->data);

q=q->next;

}

}

}

void delete\_beg()

{

if(start==NULL)

{

printf("The list is empty!!");

}

else

{

q=start;

start=start->next;

printf("Deleted element is %d",q->data);

free(q);

}

}

void delete\_end()

{

if(start==NULL)

{

printf("The list is empty!!");

}

else

{

q=start;

while(q->next->next!=NULL)

q=q->next;

t=q->next;

q->next=NULL;

printf("Deleted element is %d",t->data);

free(t);

}

}

int delete\_pos()

{

int pos,i;

if(start==NULL)

{

printf("List is empty!!");

return 0;

}

printf("Enter position to delete:");

scanf("%d",&pos);

q=start;

for(i=1;i<pos-1;i++)

{

if(q->next==NULL)

{

printf("There are less elements!!");

return 0;

}

q=q->next;

}

t=q->next;

q->next=t->next;

printf("Deleted element is %d",t->data);

free(t);

return 0;

}