Assignment 1

Data Structures

Due Date: 11th September 2017, 11:59 PM

Question 1. Devise an algorithm to determine the Nth-to-Last element in a singly linked list of unknown length. If N = 0, then your algorithm must return the last element.

#include<stdafx.h>

#include<stdio.h>

#include<assert.h>

#include<stdlib.h>

#include<iostream>

using namespace std;

struct Node

{

int data;

struct Node\* next;

};

void PrintNFromLast(struct Node\* head, int n)

{

int len = 0, i;

struct Node \*temp = head;

while (temp != NULL)

{

temp = temp->next;

len++;

}

if (len < n)

return;

temp = head;

for (i = 1; i < len - n + 1; i++)

temp = temp->next;

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

return;

}

void push(struct Node\*\* head\_ref, int new\_data)

{

struct Node\* new\_node =

(struct Node\*) malloc(sizeof(struct Node));

new\_node->data = new\_data;

new\_node->next = (\*head\_ref);

(\*head\_ref) = new\_node;

}

int main()

{

struct Node\* head = NULL;

push(&head, 12);

push(&head, 26);

push(&head, 14);

push(&head, 42);

PrintNFromLast(head, 3);

return 0;

}

Question 2. You are given a pointer to a node (not the tail node) in a singly linked list. Write an algorithm to delete that node from the linked list.

#include<stdafx.h>

#include<stdio.h>

#include<assert.h>

#include<stdlib.h>

#include<iostream>

using namespace std;

struct Node

{

int data;

struct Node\* next;

};

void push(struct Node\*\* head\_ref, int Ndata)

{

struct Node\* new\_node =

(struct Node\*) malloc(sizeof(struct Node));

new\_node->data = Ndata;

new\_node->next = (\*head\_ref);

(\*head\_ref) = new\_node;

}

void printList(struct Node \*head)

{

struct Node \*temp = head;

while (temp != NULL)

{

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

temp = temp->next;

}

}

void DeleteNode(struct Node \*pNode)

{

struct Node \*temp = pNode->next;

pNode->data = temp->data;

pNode->next = temp->next;

free(temp);

}

int main()

{

struct Node\* head = NULL;

push(&head, 15);

push(&head, 2);

push(&head, 11);

push(&head, 3);

push(&head, 5);

cout << " Before deleting " << endl;

printList(head);

DeleteNode(head);

cout << endl;

cout << " After deleting " << endl;

printList(head);

getchar();

}

Question 3. Write an algorithm to reverse singly linked list. You can assume the size of the linked list on your own however, the number of the nodes in the linked list should not be less than 5.

#include<stdafx.h>

#include<stdio.h>

#include<assert.h>

#include<stdlib.h>

#include<iostream>

using namespace std;

struct Node

{

int data;

struct Node\* next;

};

static void Reverse(struct Node\*\* head\_ref)

{

struct Node\* prev = NULL;

struct Node\* current = \*head\_ref;

struct Node\* next;

while (current != NULL)

{

next = current->next;

current->next = prev;

prev = current;

current = next;

}

\*head\_ref = prev;

}

void Push(struct Node\*\* head\_ref, int new\_data)

{

struct Node\* new\_node =

(struct Node\*) malloc(sizeof(struct Node));

new\_node->data = new\_data;

new\_node->next = (\*head\_ref);

(\*head\_ref) = new\_node;

}

void PrintList(struct Node \*head)

{

struct Node \*temp = head;

while (temp != NULL)

{

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

temp = temp->next;

}

}

int main()

{

struct Node\* head = NULL;

Push(&head, 52);

Push(&head, 48);

Push(&head, 76);

Push(&head, 63);

cout << "Given linked list" << endl;

PrintList(head);

Reverse(&head);

cout << endl;

cout << "Reversed Linked list" << endl;

PrintList(head);

getchar();

}

Question 4. How would you find a loop in a singly-linked list? Provide an elegant algorithm for this problem such that your solution should run in O(n) time and O(1) space.

**I don’t really know how to do this since we havn’t to the best of my knowledge talked about O(n) times or O(1) space, I don’t know what that means.**

Hint: The loop in the singly-linked list doesn’t have to be at the head. It can be like this for example:

A->B->C->D->E->C

You logic won’t be able to detect the loop since it will never reach the head again. Hope this helps.

Question 5. Write a theoretical solution to find the middle element of singly linked list in two pass and one pass.

**I would make a counter variable that counts the total number of nodes. If the number is odd it would divide the number of nodes by 2 then round up. If it was even it would divide by 2 and print that node since it would be the middle node**

Question 6. Write a theoretical solution for the array having 100 random integer numbers and out of those 100 one is duplicate.

Question 7. Write a theoretical solution to find a 3rd element from end in a singly linked list in one pass.

**So first I would have the program find the value of the final address then I would take that last address and subtract 2 from there to get to the third to last element**

**If that proved too hard I would reverse the linked list and count to the third element before printing that one since it would be third from the end of the original linked list.**

Question 8. Provide a design for the implementation of the Circular Singly Linked List.

