Assignment 1 – Coleman Lyski

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.

Node \*nthToLast( Node \*head, int n )

{

Node \*ptr;

Node \*back;

ptr = head;

for( int i = 0; i < n; i++ ) {

if( ptr->next != NULL ) {

ptr = ptr->next;

} else {

return NULL;

}

}

back = head;

while( ptr->next ) {

ptr = ptr->next;

back = back->next;

}

if( n == 0 ){

return ptr;

} else {

return back;

}

}

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.

void deleteNode( node \*head, int n ){

Node \*ptr, \*loc;

loc = search(head, n);

if( loc == (\*node) NULL ){

return;

}

ptr = loc->next;

loc->next = ptr->next;

free(ptr);

}

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.

void reverse(struct Node \*\*head)

{

if( head == NULL ){

return;

}

Node \*prev = NULL, \*ptr = NULL, \*next = NULL;

ptr = head;

while (ptr != NULL)

{

next = ptr->next;

ptr->next = prev;

prev = ptr;

ptr = next;

}

head = prev;

}

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.

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.

bool listLoop(struct Node \*list)

{

struct Node \*slow\_ptr = list;

struct Node \*fast\_ptr = list;

while( slow\_ptr && fast\_ptr && fast\_ptr->next ) {

slow\_ptr = slow\_ptr->next;

fast\_ptr = fast\_ptr->next->next;

if( slow\_ptr == fast\_ptr )

{

return true;

}

}

return false;

}

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

One Pass:

struct Node\* findMiddle(struct Node \*head)

{

if (head == NULL)

return NULL;

if (head->next == NULL)

{

return NULL;

}

struct Node \*slow\_ptr = head;

struct Node \*fast\_ptr = head;

while (fast\_ptr != NULL && fast\_ptr->next != NULL)

{

fast\_ptr = fast\_ptr->next->next;

slow\_ptr = slow\_ptr->next;

}

return slow\_ptr;

}

Two Passes:

Go through the entire linked list and make a variable to keep track of how many elements there are in total. Then divide that number by two and on the second pass go through the linked list to that number.

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

I’m not really sure what this question is even asking. If I were making an array with random numbers 1 to 100, I could do rand() % 100 + 1, and that could result in a duplicate.

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

Start a first pointer and let it go through three elements of the list. Once the first pointer gets to the third element, start a second pointer at the first element. When the first pointer hits a NULL, then return the second pointer.

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

Point the last element to the address of the head element. This would create a circular singly linked list.

