BONIFACE MWANGI WARIGI

REG NO:SCT212-0726/2022

UNIT: ICS 2105 DSA

QUIZ 2-LINKED LISTS

QUESTION ONE (1):

To solve this problem, we can use Floyd's cycle-finding algorithm, also known as the "tortoise and the hare" algorithm. This algorithm uses two pointers, one that moves two steps at a time and another that moves one step at a time. If there is a cycle in the list, the fast pointer (the hare) will eventually meet the slow pointer (the tortoise).

Here are the steps to solve this problem:

1. Initialize two pointers, slow and fast, to the head of the linked list.

2. While fast is not null and fast.next is not null (to avoid null pointer exception), do the following:

a. Move slow one step at a time. So, slow = slow.next.

b. Move fast two steps at a time. So, fast = fast.next.next.

c. If slow == fast, then there is a cycle in the linked list. So, return true.

3. If we exit the loop, it means that we've reached the end of the linked list and there is no cycle. So, return false.

QUESTION TWO (2):

To find the node where the cycle begins, we can still use Floyd's cycle-finding algorithm. After finding that a cycle exists (when slow == fast), we can find the node where the cycle begins.

Here are the steps to solve this problem:

1. After finding that a cycle exists, initialize another pointer, say ptr, to the head of the linked list.

2. Now, move ptr and slow one step at a time.

3. The point where ptr and slow meet is the node where the cycle begins.

QUESTION THREE (3):

To reverse a linked list, we can use a three-pointer approach. We maintain three pointers: prev, curr, and next.

Here are the steps to solve this problem:

1. Initialize three pointers: prev = null, curr = head, and next = null.

2. Iterate through the linked list until curr is not null, do the following:

a. Set next to be the next node of curr. So, next = curr.next.

b. Change the next of curr to prev. So, curr.next = prev.

c. Move prev and curr one step forward. So, prev = curr and curr = next.

3. After the loop, prev will be pointing to the new head of the reversed linked list. So, return prev.