KALYANI GOVERNMENT ENGINEERING COLLEGE

Department of Computer Application

Lab Assignment 3

Topic - Linked list

Name: Madhusudan Chand
Roll No.: 10271023013
Registration No.: 231020510013
Stream: MCA
Semester: 2^{nd}

Paper: Date Structure Lab with python
Paper Code: MCAN-291

Contents

3.1 What are the steps to inserting a new item at the head of a linked list? Write the algorithm.

Ans -

These are the following steps to insert a new item at the head of a linked list -

```
Step 1 : Declare a head pointer and make it as NULL.
Step 2 : Create a new node with the given data.
Step 3 : Make the new node points to the head node.
Step 4 : Finally, make the new node as the head node.
```

Algorithm 1: Insert at the beginning of a linked list

```
Input: PTR, VAL, HEAD
Output: Updated HEAD after insertion

    [[ Here Arr is the unsorted array
       The algorithm removes duplicates elements from Arr. ]]

if PTR = NULL then
    Write "OVERFLOW";
else
    SET NEW_NODE = PTR;
    SET PTR = PTR → NEXT;
    SET NEW_NODE → DATA = VAL;
    SET NEW_NODE → NEXT = HEAD;
    SET HEAD = NEW_NODE;
end
```

3.2 Suppose that p is a reference to an IntNode in a linked list, and it is not the tail node. What are the steps to removing the node after p? Write the algorithm..

Ans -

To remove the node after p in a linked list, you need to perform the following steps:

```
Step 1 : Check if p is not the tail node and
         p.next is not null.
Step 2 : Set temp to p.next to keep a
         reference to the node to be removed.
Step 3 : Update p.next to skip over the node to be removed,
         effectively removing it from the list.
Step 4 : Dispose of the temp node.
```

Algorithm 2: Remove the node after p in a linked list

```
Input: p (reference to an IntNode)
Output: None
if p is not the tail node and p.next is not null then
    Set temp = p.next;
    Set p.next = temp.next;
    Dispose of temp;
end
```

3.3 Suppose we are using the usual IntNode structure (with instance variables called data and link). Your program is using an IntNode variable called head to refer to the first node of a linked list (or head is null for the empty list). Write a few lines of Python code that will print all the prime numbers on the list.

Ans - Here's an example of Python code that prints all the prime numbers in a linked list of integers, where each node is an IntNode with instance variables data and link, and head is a reference to the first node of the linked list

Code :

```
class IntNode:
def __init__(self, data=0, next=None):
    self.data = data
    self.next = next

def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, int(n ** 0.5) + 1,2):
        if n % i == 0:
            return False
    return True

def print_prime_numbers(head):
    current = head
    while current is not None:
        if is_prime(current.data):
            print(current.data)
        current = current.link

head = IntNode(11, IntNode(2, IntNode(3,
IntNode(4, IntNode(5, None)))))

print_prime_numbers(head)
```

Output

```
11
2
3
5
```

3.4 Suppose we are using the usual IntNode structure, and ”locate” is referring to a node in a linked list. Write an assignment statement that will make locate refer to the next node in the list (if there is one). If there is no next node, then your assignment statement should set locate to null.

Ans - To make locate refer to the next node in the linked list (if there is one), or set it to None if there is no next node, you can use the following assignment statement in Python:

```
locate = locate.next if locate is not None else None
```

This statement checks if locate is not None. If it’s not None, it updates locate to refer to the next node by accessing the next attribute of the current node (locate.next). If locate is None, meaning there is no next node, it sets locate to None.

3.5 Implement using Python codes corresponding to the following method as a new static method for the IntNode structure.
public static int count42s(IntNode head)
// Precondition: head is the head reference of a linked list.
// The list might be empty or it might be non-empty.
// Postcondition: The return value is the number of occurrences
// of 42 in the data field of a node on the linked list.
// The list itself is unchanged.

Ans -
Code :

```
class IntNode:
    def __init__(self, data=0, next=None):
        self.data = data
        self.next = next

    @staticmethod
    def count42s(head):
        count = 0
        current = head
        while current is not None:
            if current.data == 42:
                count += 1
            current = current.next
        return count

head = IntNode(1, IntNode(42, IntNode(3,
IntNode(42, IntNode(5, None)))))
print("Number of 42s ",IntNode.count42s(head))
```

Output

```
Number of 42s  2
```

3.6 Implement using Python codes corresponding to the following method as a new static method for the IntNode structure.
public static boolean has42(IntNode head)
// Precondition: head is the head reference of a linked list.
// The list might be empty or it might be non-empty.
// Postcondition: The return value is true if the list has at least
// one occurrence of the number 42 in the data part of a node.

Ans -

Code

```
class IntNode:
    def __init__(self, data=0, next=None):
        self.data = data
        self.next = next

    @staticmethod
    def count42s(head):
        current = head
        while current is not None:
            if current.data == 42:
                return True
            current = current.next
        return False

head = IntNode(1, IntNode(42, IntNode(3,
IntNode(42, IntNode(5, None)))))
print("At least one occurrence of the number 42",
IntNode.count42s(head))
```

Output:

```
At least one occurrence of the number 42  True
```

3.7 Implement using Python codes corresponding to the following method as a new method for the IntNode structure.
public static boolean all42s(IntNode head)
// Precondition: head is the head reference of a linked list.
// The list might be empty or it might be non-empty.
// Postcondition: The return value is true if every node in the
// list contains 42 in the data part. NOTE: If the list is empty,
// then the method returns true.

Ans -

Code :

```
class IntNode:
    def __init__(self, data=0, next=None):
        self.data = data
        self.next = next


    @staticmethod
    def count42s(head):
        current = head
        while current is not None:
            if current.data != 42:
                return False
            current = current.next
        return True
    def print_list(head):
        current = head
        while current is not None:
            print(current.data, end=" ")
            current = current.next
        print()

head = IntNode(42, IntNode(42, IntNode(42,
IntNode(42, IntNode(42, None)))))
print("The list")
IntNode.print_list(head)
print("Every node in list contain 42",IntNode.count42s(head))
```

Output

```
The list
42 42 42 42 42
Every node in list contain 42 True
```