Linked List

د. عامر جعفر صادق د. هاله حسن محمود م.م امنه هيثم عبد اللطيف

Q1) implement Linked list operations in Python

Program

```
# Create a node
class Node:
    def __init__(self, item):
        self.item = item
        self.next = None

class LinkedList:
    def __init__(self):
        self.head = None

    # Insert at the beginning
    def insertAtBeginning(self, data):
        new_node = Node(data)
        new_node.next = self.head
        self.head = new_node

    # Insert after a node
    def insertAfter(self, node, data):

        if node is None:
            print("The given previous node must inLinkedList.")
            return

        new_node = Node(data)
        new_node.next = node.next
        node.next = new_node

    # Insert at the end
    def insertAtEnd(self, data):
        new_node = Node(data)
```

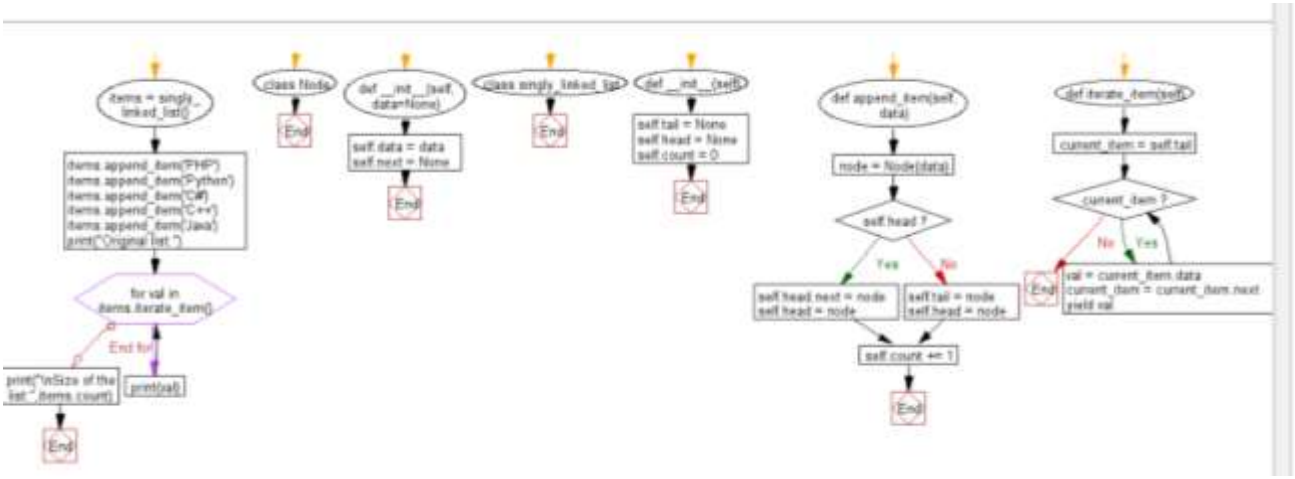
```
    if self.head is None:
        self.head = new_node
        return
    last = self.head
    while (last.next):
        last = last.next

    last.next = new_node
# Deleting a node
def deleteNode(self, position):
    if self.head == None:
        return
    temp_node = self.head
    if position == 0:
        self.head = temp_node.next
        temp_node = None
        return

    # Find the key to be deleted
    for i in range(position - 1):
        temp_node = temp_node.next
        if temp_node is None:
            break

    # If the key is not present
    if temp_node is None:
        return

    if temp_node.next is None:
        return

    next = temp_node.next.next
    temp_node.next = None
    temp_node.next = next

def printList(self):
    temp_node = self.head
    while (temp_node):
        print(str(temp_node.item) + " ", end="")
        temp_node = temp_node.next
```

```
if __name__ == '__main__':

    llist = LinkedList()
    llist.insertAtEnd(1)
    llist.insertAtBeginning(2)
    llist.insertAtBeginning(3)
    llist.insertAtEnd(4)
    llist.insertAfter(llist.head.next, 5)

    print('Linked list:')
    llist.printList()

    print("\nAfter deleting an element:")
    llist.deleteNode(3)
    llist.printList()
```

Output

```
Linked list:
3 2 5 1 4
After deleting an element:
3 2 5 4
```

Q2) Write a Python program to find the size of a singly linked list.



Program

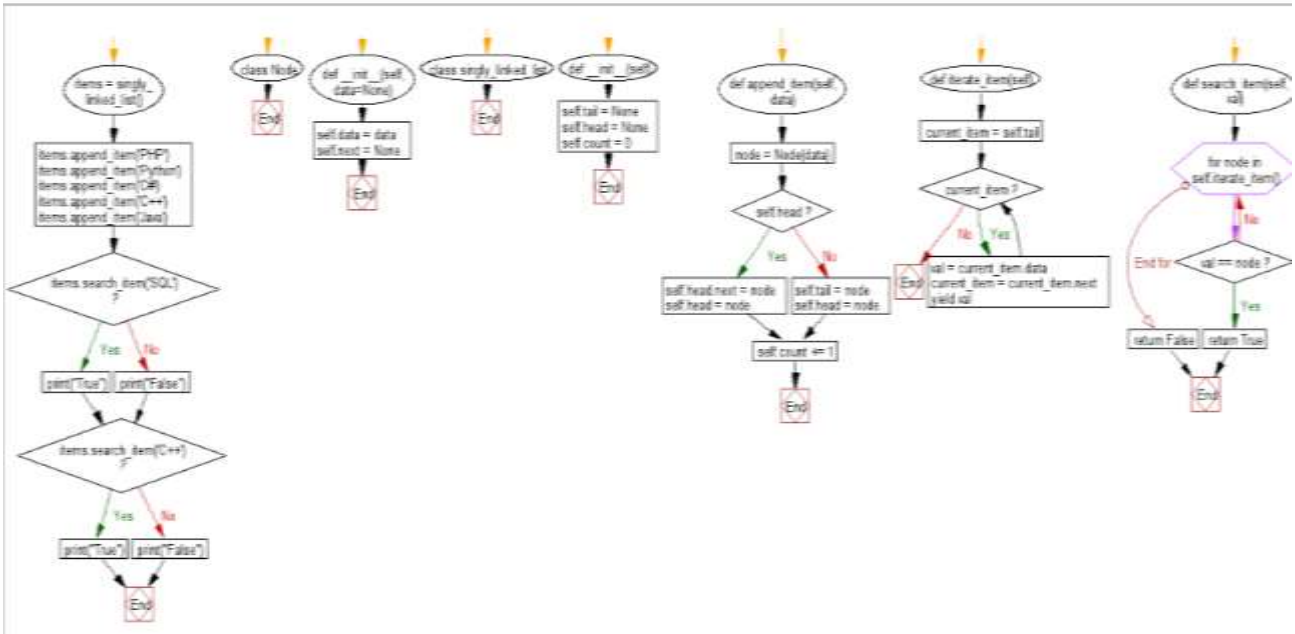
```
class Node:
  # Singly linked node
  def __init__(self, data=None):
     self.data = data
     self.next = None
class singly_linked_list:
  def __init__(self):
     # Createe an empty list
     self.tail = None
     self.head = None
     self.count = 0

  def append_item(self, data):
     #Append items on the list
     node = Node(data)
     if self.head:
        self.head.next = node
        self.head = node
     else:
        self.tail = node
```

```
            self.head = node
        self.count += 1

    def iterate_item(self):
        # Iterate the list.
        current_item = self.tail
        while current_item:
            val = current_item.data
            current_item = current_item.next
            yield val

items = singly_linked_list()
items.append_item('PHP')
items.append_item('Python')
items.append_item('C#')
items.append_item('C++')
items.append_item('Java')

print("Original list:")
for val in items.iterate_item():
    print(val)

print("\nSize of the list:",items.count)
```

Output

```
Original list:
PHP
Python
C#
C++
Java

Size of the list: 5
```

Q3) Write a Python program to search a specific item in a singly linked list and return true if the item is found otherwise return false.



Program

```
class Node:
    # Singly linked node
    def __init__(self, data=None):
        self.data = data
        self.next = None
class singly_linked_list:
    def __init__(self):
        # Createe an empty list
        self.tail = None
        self.head = None
        self.count = 0

    def append_item(self, data):
        #Append items on the list
        node = Node(data)
        if self.head:
            self.head.next = node
            self.head = node
```

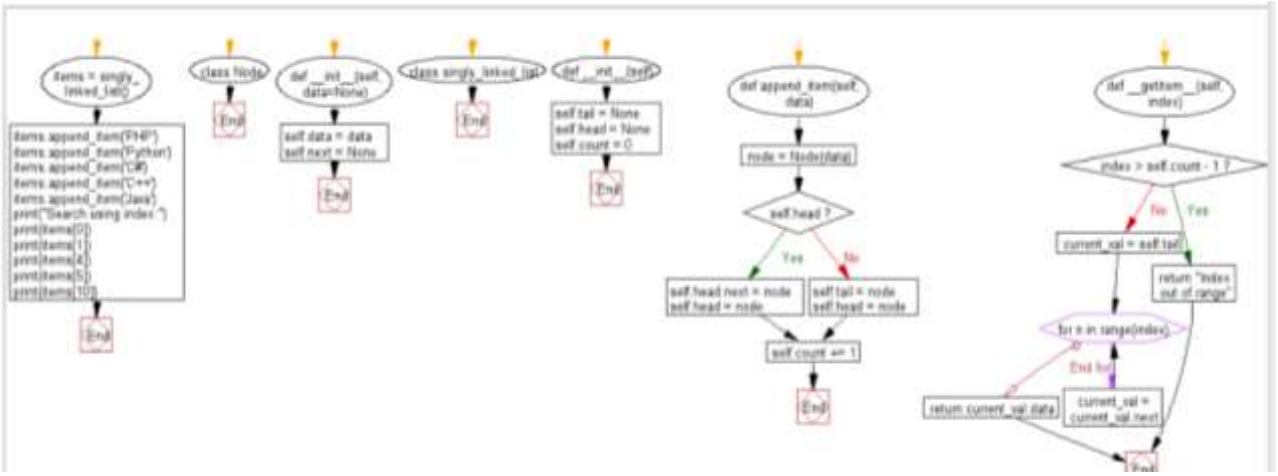
```
            else:
                self.tail = node
                self.head = node
            self.count += 1

        def iterate_item(self):
            # Iterate the list.
            current_item = self.tail
            while current_item:
                val = current_item.data
                current_item = current_item.next
                yield val

        def search_item(self, val):
            # Search the list
            for node in self.iterate_item():
                if val == node:
                    return True
            return False
items = singly_linked_list()
items.append_item('PHP')
items.append_item('Python')
items.append_item('C#')
items.append_item('C++')
items.append_item('Java')
if items.search_item('SQL'):
    print("True")
else:
    print("False")
if items.search_item('C++'):
    print("True")
else:
    print("False")
```

Output

```
False
True
```

د. عامر جعفر صادق د. هاله حسن محمود م.م امنه هيثم عبد اللطيف

Q4) Write a Python program to access a specific item in a singly linked list using index value.



Program

```
class Node:
    # Singly linked node
    def __init__(self, data=None):
        self.data = data
        self.next = None
class singly_linked_list:

    def __init__(self):
        # Createe an empty list
        self.tail = None
        self.head = None
        self.count = 0

    def append_item(self, data):
        #Append items on the list
        node = Node(data)
        if self.head:
            self.head.next = node
            self.head = node
        else:
            self.tail = node
            self.head = node
        self.count += 1
```

```
    def __getitem__(self, index):
        if index > self.count - 1:
            return "Index out of range"
        current_val = self.tail
        for n in range(index):
            current_val = current_val.next
        return current_val.data


items = singly_linked_list()
items.append_item('PHP')
items.append_item('Python')
items.append_item('C#')
items.append_item('C++')
items.append_item('Java')

print("Search using index:")
print(items[0])
print(items[1])
print(items[4])
print(items[5])
print(items[10])

print(q.dequeue())
```

Output

```
Search using index:
PHP
Python
Java
Index out of range
Index out of range
```