정보구조 과제 6
2014-19498 정은주
독어교육과

problem 1

In [1]:
```
#problem 1
```

In [2]:
```
def search(H, value):
    while(H is not None):
        if H.element == value:
            return True
        H = H.next
    return False
```

In [3]:
```
class Node:
    def __init__(self, element, next = None):
        self.element = element
        self.next = next
```

In [4]:
```
S1 = Node(1)
```

In [5]:
```
S2 = Node(2)
```

In [6]:
```
S3 = Node(3)
```

In [7]:
```
S1.next = S2
S2.next = S3
```

In [8]:
```
search(S1, 3)
```

Out[8]:
```
True
```

In [9]:
```
search(S1, 4)
```

Out[9]:
```
False
```

Problem 2

1번 문제의 node s1, s2, s3을 이용, s1(1)+s2(2)+s3(3) = 6

In [11]:
```
def sum_list(H):
    sum = 0
    while(H is not None):
        sum += H.element
        H = H.next
    return sum
```

In [12]:
```
sum_list(S1)
```

Out[12]: 6

Problem 3

In [14]:
```
#Problem 3
```

In [171]:
```
class Empty(Exception):
    pass

class LinkedQueue:
    class _Node:
        def __init__(self, element = None, next = None):
            self._element = element
            self._next = next

    def __init__(self):
        self._size = 0
        self._header = self._Node()
        self._trailer = self._Node()
        self._header._next = self._trailer

    def __len__(self):
        return self._size

    def is_empty(self):
        return self._size == 0

    def first(self):
        if self.is_empty():
            raise Empty('Queue is empty')
        return self._header._element

    def enqueue(self, e):
        node = self._Node(e)
        if self.is_empty():
            self._header = node

        else:
            self._trailer._next = node

        self._trailer = node
        self._size += 1

    def dequeue(self):
        if self.is_empty():
            raise Empty('Queue is empty')
        val = self._header._element
        self._header = self._header._next
        self._size -=1
        if self.is_empty():
            self._trailer = None
        return val
```

In [172]:
```
Q = LinkedQueue()
```

In [ ]:
```

```

Problem 4

함수 구현이 맞는지 확인하기 위해서 DoublyLinkedList class에 list() 함수를 추가하였다. 본래 doubly linked를 약간 변형하여 insert를 trailer 위치에서 가능하도록 했다. 2개의 linked list를 만들기 용이하도록 그리고 본 문제의 초점인 concatenate_list 함수 확인의 간편함을 위해서다.

In [88]:
```
#problem 4
```

In [402]:
```
class DoublyLinkedList:
    class _Node:
        def __init__(self, element = None, prev = None, next = None):
            self._element = element
            self._prev = prev
            self._next = next

    def __init__(self):
        self.header = self._Node()
        self.trailer = self._Node()
        self.header._next = self.trailer
        self.trailer._prev = self.header
        self._size = 0

    def __len__(self):
        return self._size

    def is_empty(self):
        return self._size == 0

    def insert(self, e):
        node = self._Node(e)
        self.trailer._next = node
        node._prev = self.trailer
        self.trailer = node
        self._size += 1
        return node._element

    def list(self):
        node = self.header
        list = []
        while(node is not None):
            if(node._element is not None):
                list.append(node._element)
            node = node._next
        return list
```

In [403]:
```
L1 = DoublyLinkedList()
```

In [404]:
```
L1.insert(1)
```

Out[404]: 1

In [405]:
```
L1.insert(2)
```

Out[405]: 2

In [406]:
```
L1.insert(3)
```

Out[406]: 3

In [407]:
```
L1.insert(4)
```

Out[407]: 4

In [408]:
```
L1.insert(5)
```

Out[405]: 2

In [406]:
```
L1.insert(3)
```

Out[406]: 3

In [407]:
```
L1.insert(4)
```

Out[407]: 4

In [408]:
```
L1.insert(5)
```

Out[408]: 5

In [409]:
```
len(L1)
```

Out[409]: 5

In [410]:
```
L2 = DoublyLinkedList()
```

In [411]:
```
L2.insert(6)
```

Out[411]: 6

In [412]:
```
L2.insert(7)
```

Out[412]: 7

In [413]:
```
L2.insert(8)
```

Out[413]: 8

In [414]:
```
L2.insert(9)
```

Out[414]: 9

In [415]:
```
def concatenate_list(H1, T1, H2, T2):
    L3 = DoublyLinkedList()
    T1._next = H2
    H2._prev = T1
    L3.header = H1
    L3.trailer = T2
    return L3.list()
```

In [416]:
```
concatenate_list(L1.header, L1.trailer, L2.header, L2.trailer)
```

Out[416]: [1, 2, 3, 4, 5, 6, 7, 8, 9]

In [ ]: