Group Activity 03

(3인 혹은 4인으로 팀을 구성하여 아래의 문제를 푼다. 팀 구성은 매 시간마다 달라져도 된다.)

팀원1: _____

팀원2: _____

팀원3: _____

팀원4: _____

1. 입력으로 두 개의 연결리스트를 받아서 첫 번째 연결리스트의 맨 끝에 두 번째 연결리스트를 연결하여 하나의 연결리스트로 합치는 일을 하는 함수 `concatenate`를 작성하라. 새롭게 만들어진 연결리스트의 head 노드의 주소를 반환해야 한다. 입력으로 들어온 연결리스트들은 empty list일 수도 있다.

```
Node *concatenate(Node *first, Node *second) {









}
```

2. 다음의 함수는 각 노드에 하나의 정수를 저장하는 연결리스트에서 처음으로 등장하는 3의 배수를 (만약 존재한다면) 연결리스트로 부터 삭제하는 일을 하려고 작성된 것이다. 이 함수는 의도한 일을 제대로 수행하는가? 어떤 문제가 있는가? 만약 문제가 있다면 어떻게 수정해야 할까? 수정된 함수를 어떻게 호출해야 할까?

```
struct Node  {
    int data;
    Node *next;
};

void remove_multiple_three(Node *head)  {
    struct node *p = head;
    while (p->next->data % 3 != 0)
        p = p->next;
    if (p->next == nullptr)
        return;
    p->next = p->next->next;
}
```

3. 아래의 함수 func가 하려는 일은 무엇일까? 이 함수는 의도한 일을 제대로 수행하는가? 만약 그렇지 않다면 어떻게 수정해야 할까? 수정된 함수를 어떻게 호출해야 할까?

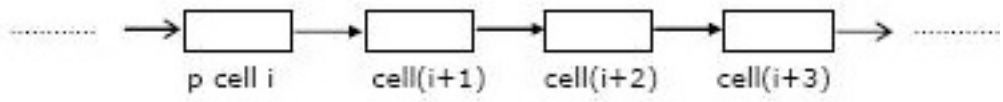
```
struct Node {
    int data;
    Node *next;
};

void func(Node* head)  {
    Node* prev = nullptr;
    Node* current = head;
    Node* next;
    while (current != nullptr)  {
        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
}
```

4. 단방향 연결리스트의 첫 번째 노드와 마지막 노드의 주소를 가지고 있다. 다음 중 연결리스트의 노드의 개수에 비례하는 시간이 필요한 일은? 이유는?

(1) 첫 번째 노드 삭제하기
(2) 맨 앞에 새로운 노드 삽입하기
(3) 마지막 노드 삭제하기
(4) 마지막에 새로운 노드 삽입하기

5. 연결리스트의 마지막 노드를 연결 리스트의 맨 앞으로 이동시키는 일을 하는 함수를 구현하라. 다른 노드들의 순서가 변경되어서는 안된다. 이 일을 하기 위해서 새로운 노드를 생성해서는 안된다. 함수의 매개변수, 반환값, 함수를 호출하는 방식은 적절하게 지정하라.

6. 아래의 그림과 같은 상태이다. 이 상태에서 다음의 코드를 실행한 결과를 그림으로 표현하라.




```
q = p→next
p→next = q→next
q→next = q→next→next
p→next→next = q
```

7. 다음의 함수 rearrange는 정수들이 저장된 하나의 연결리스트를 매개변수로 받는다. 정수 1, 2, 3, 4, 5, 6, 7이 순서대로 저장된 연결리스트가 입력 매개변수로 주어졌다면 함수가 종료한 후에 연결리스트에 저장된 정수들의 순서는?

```
struct Node  {
  int value;
  Node *next;
};

void rearrange(Node *list)  {
  Node *p, *q;
  int temp;
  if (list==nullptr || list->next==nullptr)
      return;
  p = list;
  q = list->next;
  while(q!=nullptr)  {
     temp = p->value;
     p->value = q->value;
     q->value = temp;
     p = q->next;
     q = (p != nullptr ? p->next : 0);
  }
}
```