실습과제 3.3

[과제 설명]

연결 리스트가 역순으로 출력되게 구성한다. 원본 연결 리스트의 시작 주소를 저장하고 있는 head는 바꾸지 않고 새로운 연결 리스트를 하나 더 만들어서 시작 주소를 반환하기 때문에 반환 타입은 Node*이고, 인수로는 더블 포인터가 아닌 Node*로 받는다.
먼저 임시 저장소가 될 노드를 2개 더 선언해준다. 그러고 아래의 과정을 p가 NULL이 될 때까지 계속 반복한다.

1. q를 r에 대입한다.
2. p를 q에 대입한다.
3. p를 다음 노드로 이동한다.
4. q의 link 부분에 r의 주소를 넣는다.

이 과정이 반복되고 나면 q에 역순으로 정렬된 연결 리스트가 생성된다. 그렇기 때문에 q의 주소를 반환한다.

[코드]

헤더 파일을 분리할 정도로 함수를 많이 사용하지 않았기도 하고 보고서를 작성할 때 필요 이상으로 길어지는 것을 방지하기 위해 이번 실습에 쓰지 않은 함수를 제외하고 한 코드로 첨부한다.

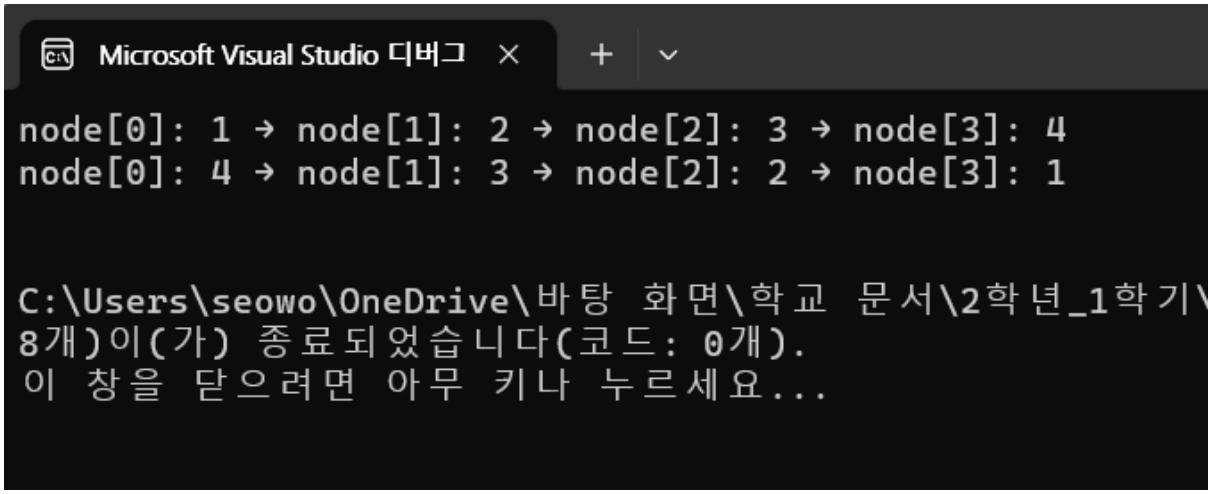
또한 이전 실습에서도 사용한 함수는 주석을 생략한다.

```
#include <stdio.h>
#include <stdlib.h>

typedef struct Node {
```

```
  int data;
  struct Node* link;
}Node;

Node* Create_Node(int newData){
  Node* newNode = (Node*)malloc(sizeof(Node));
  newNode->data = newData;
  newNode->link = NULL;

  return newNode;
}
void Append_Node(Node** head, Node* newNode) {
  if ((*head) == NULL)
    *head = newNode;
  else {
    Node* tail = (*head);
    while (tail->link != NULL) {
      tail = tail->link;
    }
    tail->link = newNode;
  }
}
void Print_Liked_List(Node* head) {
  Node* iter = head;
  int i = 0;
  while (iter != NULL) {
    printf("node[%d]: %d", i, iter->data);
    iter = iter->link;
    if (iter != NULL) printf(" → ");
    i++;
  }
  puts("");
}
Node* Reverse_List(Node* head) {
  Node* p = head;
  Node* q = NULL;
  Node* r;

  while (p != NULL) {
    r = q;
    q = p; //r에 q를, p를 q에 복사하고
    p = p->link; //p를 하나 이동
    q->link = r; //그리고 q의 link에 r을 붙인다.
  }//p가 더 없을 때까지 반복

  return q;
  //역순으로 정렬된 새로운 연결 리스트 q의 주소를 반환한다.
}

int main(void) {
  Node* head = NULL;

  //기본 연결리스트 구성
  Append_Node(&head, Create_Node(1));
  Append_Node(&head, Create_Node(2));
  Append_Node(&head, Create_Node(3));
  Append_Node(&head, Create_Node(4));
```

```
    //기본 연결리스트 출력
    Print_Liked_List(head);

    //역 연결리스트 함수 호출
    Node* reverse = Reverse_List(head);
    //역 연결리스트 출력
    Print_Liked_List(reverse);
    puts("");
}
```

[실행 결과]



[과제에 대한 고찰]

연결 리스트를 역순으로 정렬할 때는 단순히 temp 하나만 이용해서 역순으로 만들 수는 없었다. 다음 노드의 link를 저장하는 특성 때문이었다. 그렇다고 해도 의사 코드와 알고리즘이 미리 교안에 있어서 그런지 생각보다 어렵지는 않았던 것 같다.

이걸 앞의 (출력 함수를 포함한) 7개의 함수처럼 많이 쓰이는 함수일까 싶어서 찾아봤는데, 주로 알고리즘 문제로 나온다는 글이 많았고 실제로 이것이 프로그램에서 많이 쓰이는 것 같지는 않았던 것 같다. 그래도 생소하고 특이한 문제였어서 흥미로웠던 것 같다.