Group Activity 04

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

팀원1: _____

팀원2: _____

팀원3: _____

팀원4: _____

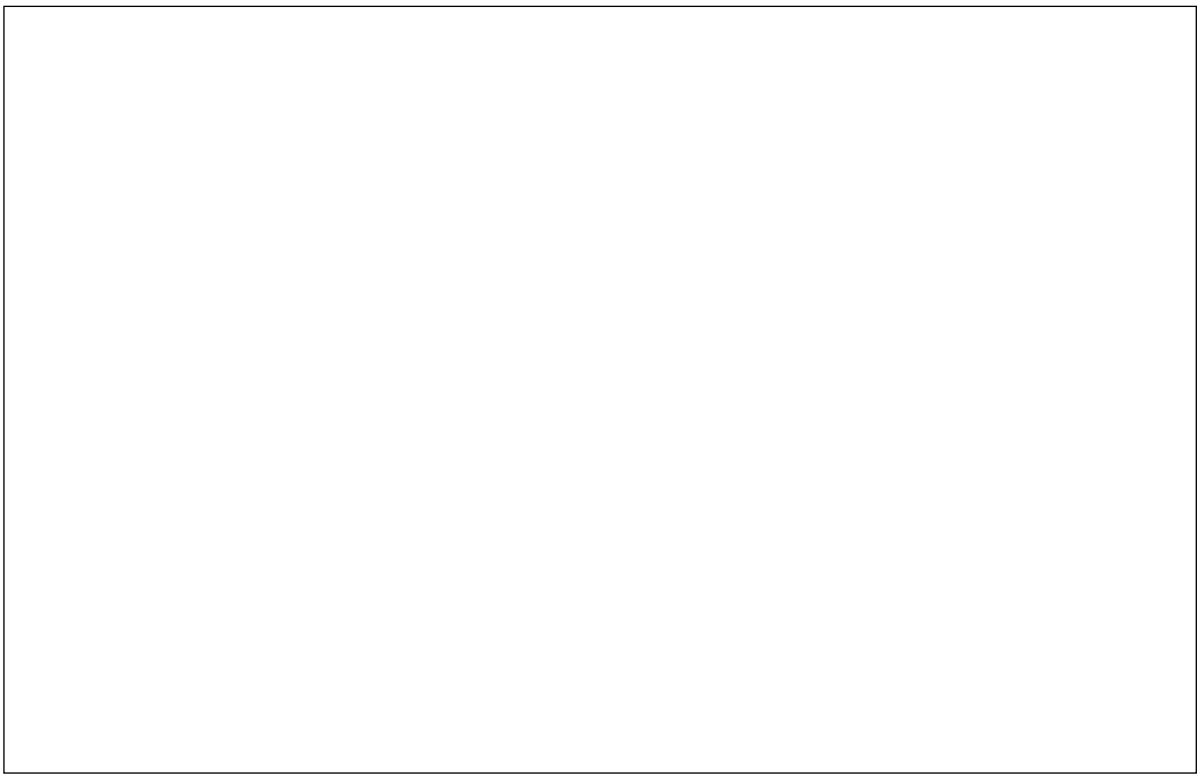
1. 정수들이 오름차순으로 정렬되어 저장된 연결리스트가 입력으로 주어질 때 아래의 함수가 하는 일은 무엇인가? 매개변수 `j`의 역할은? 그리고 연결리스트의 길이가 $n \geq 2$일 때 문장 `(S1)`과 `(S2)`가 실행되는 횟수의 최대값은 각각 얼마인가?

```
Node *func(Node *head, int *j) {
    Node *t1, *t2;
    *j=0;
    t1 = head;
    if (t1 != nullptr) t2 = t1→next;
    else return head;
    *j = 1;
    if (t2 == nullptr)
        return head;
    while (t2 != nullptr) {
        if (t1->data != t2->data)  {              // (S1)
            (*j)++; t1->next = t2; t1 = t2;       // (S2)
        }
        t2 = t2→next;
    }
    t1→next = nullptr;
    return head;
}
```

2. 정수들이 오름차순으로 정렬되어 저장되어 있는 2개의 연결리스트를 받아서 두 연결리스트를 하나의 정렬된 연결리스트로 합친 후 합쳐진 연결리스트의 첫 번째 노드의 주소를 반환하는 함수

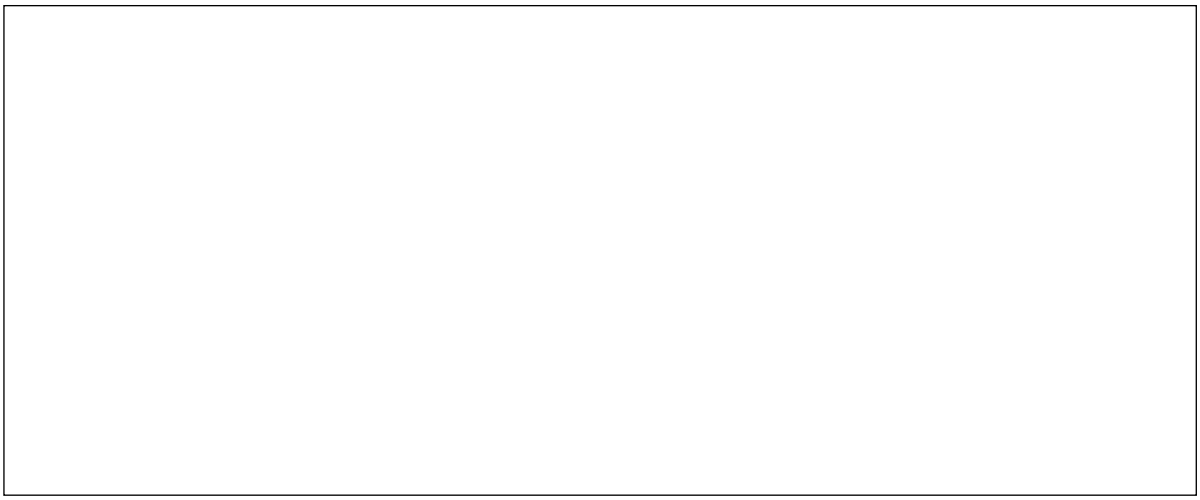
```
Node *merge(Node *first, Node *second)
```

를 작성하라. 새로운 노드를 생성해서는 안된다.



3. 정수들이 저장된 연결리스트를 매개변수로 받는 다음의 함수가 하려는 일은 무엇일까? 문제가 없는가? 문제가 있다면 어떤 문제인지 밝히고 수정하라.

```
void func(Node *head)  {
    Node *p = head, *q = nullptr;
    while (p != nullptr) {
        if (p->data < 0)
            q->next = p->next;
        else
            q = p;
        p = p->next;
    }
}
```



4. 오름차순으로 정렬된 정수들이 저장된 연결리스트와 두 정수 lower와 upper가 매개변수로 주어진다. lower보다 크거나 같고 upper보다 작거나 같은 정수를 저장하는 모든 노드를 연결리스트로부터 삭제하는 일을 하는 함수를 작성하라. 단, lower ≤ upper이다.

5. 다음 프로그램들의 출력은? 컴파일 오류나 실행 오류가 나는 경우에는 이유를 간략히 설명하라.

Program	Output

#include <iostream>
#include <list>
using namespace std;
int main() {
  list<int> first;
  list<int> second(4,100);
  list<int> third(second.begin(), second.end());
  list<int> fourth(third);
 	
  int myints[] = {16,2,77,29};
  list<int> fifth(myints, myints+sizeof(myints)/
sizeof(int));
 	
  for (auto it=fifth.begin(); it!=fifth.end(); it++)
  cout << *it << ' ';
  cout << '\n';
  return 0;
}	

```cpp
#include <iostream>
#include <list>
using namespace std;

int main () {
    list<int> first {1, 2, 3};
    list<int> second{10, 20, 30, 40, 50};

    second = first;
    auto it = first.begin();
    *it = 100;
    for (auto a: first)
        cout << a << ' ';
    cout << endl;
    for (auto b: second)
        cout << b << ' ';
    cout << endl;
    for (auto &c: second)
        c += 1;
    for (auto c: second)
        cout << c << ' ';
    cout << endl;
    return 0;
}
```

```cpp
#include <iostream>
#include <list>
using namespace std;

int main () {
    list<int> first;
    list<int> second;

    first.assign(5, 100);
    second.assign(first.begin(),first.end());

    int myints[] = {17, 7, 4};
    first.assign(myints, myints+3);

    for (auto a: first)
        cout << a << ' ';
    cout << endl;

    for (auto b: second)
        cout << b << ' ';
    cout << endl;
    return 0;
}
```

```cpp
#include <iostream>
#include <list>
using namespace std;

int main () {
    list<int> mylist(2, 100);
    mylist.push_front(200);
    mylist.push_front(300);
    for (auto it=mylist.begin(); it!=mylist.end(); ++it)
        cout << ' ' << *it;

    cout << '\n';
    return 0;
}
```

```cpp
#include <iostream>
#include <list>
using namespace std;

int main () {
  list<int> mylist;
  mylist.push_back(100);
  mylist.push_back(200);
  mylist.push_back(300);
  while (!mylist.empty()) {
    cout << ' ' << mylist.front();
    mylist.pop_front();
  }
  return 0;
}
```

```cpp
#include <iostream>
#include <list>
#include <vector>
using namespace std;

int main () {
  list<int> mylist;
  for (int i=1; i<=5; ++i)
     mylist.push_back(i);

  list<int>::iterator it = mylist.begin();
  ++it;

  mylist.insert(it,10);
  mylist.insert(it,2,20);
  --it;

  vector<int> myvector(2,30);
  mylist.insert (it, myvector.begin(), myvector.end());

  for (it=mylist.begin(); it!=mylist.end(); ++it)
    cout << ' ' << *it;
  cout << '\n';
  return 0;
}
```

```cpp
#include <iostream>
#include <list>
using namespace std;
int main () {
  list<int> mylist;
  list<int>::iterator it1, it2;

  for (int i=1; i<10; ++i)
     mylist.push_back(i*10);

  it1 = it2 = mylist.begin();
  advance(it2, 6);
  ++it1;

  it1 = mylist.erase(it1);
  it2 = mylist.erase(it2);
  ++it1;
  --it2;
  mylist.erase(it1, it2);

  for (it1=mylist.begin(); it1!=mylist.end(); ++it1)
    cout << ' ' << *it1;
  cout << '\n';
  return 0;
}
```

```cpp
#include <iostream>
#include <list>
using namespace std;
int main () {
  list<int> first(3,100);
  list<int> second(5,200);
  first.swap(second);

  for (auto it=first.begin(); it!=first.end(); it++)
    cout << ' ' << *it;
  cout << '\n';

  for (auto it=second.begin(); it!=second.end(); it++)
    cout << ' ' << *it;
  cout << '\n';

  return 0;
}
```

```cpp
#include <iostream>
#include <list>

int main () {
  std::list<int> mylist1, mylist2;
  std::list<int>::iterator it;

  for (int i=1; i<=4; ++i)
     mylist1.push_back(i);

  for (int i=1; i<=3; ++i)
     mylist2.push_back(i*10);

  it = mylist1.begin();
  ++it;

  mylist1.splice (it, mylist2);
  mylist2.splice (mylist2.begin(),mylist1, it);
  it = mylist1.begin();
  std::advance(it,3);

  mylist1.splice (mylist1.begin(), mylist1, it,
                  mylist1.end());

  for (it=mylist1.begin(); it!=mylist1.end(); ++it)
    std::cout << ' ' << *it;
  std::cout << '\n';

  for (it=mylist2.begin(); it!=mylist2.end(); ++it)
    std::cout << ' ' << *it;
  std::cout << '\n';

  return 0;
}
```

```cpp
#include <iostream>
#include <list>
using namespace std;

int main () {
  int myints[]= {17,89,7,14};
  list<int> mylist (myints, myints+4);

  mylist.remove(89);

  for (auto it=mylist.begin(); it!=mylist.end(); ++it)
    cout << ' ' << *it;
  cout << '\n';
  return 0;
}
```

```cpp
#include <iostream>
#include <list>
using namespace std;

bool single_digit (const int& value) {
   return (value<10);
}

bool is_odd(const int& value) {
   return (value%2)==1;
}

int main () {
  int myints[]= {15,36,7,17,20,39,4,1};
  list<int> mylist (myints, myints+8);

  mylist.remove_if(single_digit);
  mylist.remove_if(is_odd);

  for (auto it=mylist.begin(); it!=mylist.end(); ++it)
    cout << ' ' << *it;
  cout << '\n';
  return 0;
}
```

```cpp
#include <iostream>
#include <cmath>
#include <list>
using namespace std;

bool same_integral_part(double first, double second) {
  return int(first)==int(second);
}

struct is_near {
  bool operator() (double first, double second)
  { return (fabs(first-second)<5.0); }
};

int main () {
  double mydoubles[]={ 12.15,  2.72, 73.0,  12.77,  3.14,
                       12.77, 73.35, 72.25, 15.3,
72.25 };
  list<double> mylist(mydoubles, mydoubles+10);

  mylist.sort();
  mylist.unique();
  mylist.unique(same_integral_part);
  mylist.unique(is_near());

  for (auto it=mylist.begin(); it!=mylist.end(); ++it)
    cout << ' ' << *it;
  cout << '\n';

  return 0;
}
```

```cpp
#include <iostream>
#include <list>

bool mycomparison(double first, double second)  {
  return (int(first)<int(second));
}

int main () {
  std::list<double> first, second;

  first.push_back (3.1);
  first.push_back (2.2);
  first.push_back (2.9);

  second.push_back (3.7);
  second.push_back (7.1);
  second.push_back (1.4);

  first.sort();
  second.sort();

  first.merge(second);
  second.push_back (2.1);
  first.merge(second, mycomparison);

  for (auto it=first.begin(); it!=first.end(); ++it)
    std::cout << ' ' << *it;
  std::cout << '\n';

  return 0;
}
```

```cpp
#include <iostream>
#include <list>

int main () {
  std::list<int> mylist;
  for (int i=1; i<10; ++i)
     mylist.push_back(i);

  mylist.reverse();
  for (auto it=mylist.begin(); it!=mylist.end(); ++it)
    std::cout << ' ' << *it;
  std::cout << '\n';
  return 0;
}
```

```cpp
#include <iostream>
#include <list>
#include <string>
#include <cctype>
using namespace std;

bool compare_nocase(const string& first,
                    const string& second) {
  unsigned int i=0;
  while (i<first.length() && i<second.length()) {
    if (tolower(first[i])<tolower(second[i]))
       return true;
    else if (tolower(first[i])>tolower(second[i]))
       return false;
    ++i;
  }
  return first.length() < second.length();
}

int main () {
  list<string> mylist;
  list<string>::iterator it;
  mylist.push_back("one");
  mylist.push_back("two");
  mylist.push_back("Three");
  mylist.sort();

  for (it=mylist.begin(); it!=mylist.end(); ++it)
    std::cout << ' ' << *it;
  std::cout << '\n';

  mylist.sort(compare_nocase);
  for (it=mylist.begin(); it!=mylist.end(); ++it)
    std::cout << ' ' << *it;
  std::cout << '\n';

  return 0;
}
```