Object-Oriented Programming Report

Assignment 3-2

Professor	Donggyu Sim
Department	Computer engineering
Student ID	2022202061
Name	Seoeun Yang
Class (Design / Laboratory)	1 / B (미수강시 0로 표기)
Submission Date	2023. 5. 11

Program 1

❑ 문제 설명

16 개의 숫자를 내림차순으로 정렬하고 binary search algorithm 을 사용해 특정 숫자를 찾아 반환하는 문제이다. 연결 리스트로 구현하며 0~200 사이의 숫자 16 개를 연결리스트로 연결하고 Binary Search 함수를 호출한다. 함수 내에서 insertion sort 를 사용해 노드들을 내림차순으로 정렬한다. 그 후 binary search algorithm 을 사용해 특정 숫자를 검색한다. 일치하는 숫자가 있다면 해당 노드를 반환하고 숫자가 없다면 그 숫자와 가장 가까운 수를 가지고 있는 노드를 반환한다.

❑ 결과 화면

```
Binary before insertion order :
20  65  77  159  149  142  147  118  26  172  192  79  85  63  176  111

enter the number that you want to search : 26
Result :
Insertion sort :
65  20  77  159  149  142  147  118  26  172  192  79  85  63  176  111
77  65  20  159  149  142  147  118  26  172  192  79  85  63  176  111
159  77  65  20  149  142  147  118  26  172  192  79  85  63  176  111
159  149  77  65  20  142  147  118  26  172  192  79  85  63  176  111
159  149  142  77  65  20  147  118  26  172  192  79  85  63  176  111
159  149  147  142  77  65  20  118  26  172  192  79  85  63  176  111
159  149  147  142  118  77  65  20  26  172  192  79  85  63  176  111
159  149  147  142  118  77  65  26  20  172  192  79  85  63  176  111
172  159  149  147  142  118  77  65  26  20  192  79  85  63  176  111
192  172  159  149  147  142  118  77  65  26  20  79  85  63  176  111
192  172  159  149  147  142  118  79  77  65  26  20  85  63  176  111
192  172  159  149  147  142  118  85  79  77  65  26  20  63  176  111
192  172  159  149  147  142  118  85  79  77  65  63  26  20  176  111
192  176  172  159  149  147  142  118  85  79  77  65  63  26  20  111
192  176  172  159  149  147  142  118  111  85  79  77  65  63  26  20
Binary after insertion sort :
192   176   172   159   149   147   142   118   111   85   79   77   65   63   26   20

118 -> 77 -> 63 -> 26 ->
Binary search succesful! Node num : 15
26
```

노드 순서와 검색한 숫자를 출력한다.

```
157  183  125  35  142  18  200  146  50  127  196  32  147  69  153  43

enter the number that you want to search : 65
Result :
Insertion sort :
183  157  125  35  142  18  200  146  50  127  196  32  147  69  153  43
183  157  142  125  35  18  200  146  50  127  196  32  147  69  153  43
200  183  157  142  125  35  18  146  50  127  196  32  147  69  153  43
200  183  157  146  142  125  35  18  50  127  196  32  147  69  153  43
200  183  157  146  142  125  50  35  18  127  196  32  147  69  153  43
200  183  157  146  142  127  125  50  35  18  196  32  147  69  153  43
200  196  183  157  146  142  127  125  50  35  18  32  147  69  153  43
200  196  183  157  146  142  127  125  50  35  32  18  147  69  153  43
200  196  183  157  147  146  142  127  125  50  35  32  18  69  153  43
200  196  183  157  147  146  142  127  125  69  50  35  32  18  153  43
200  196  183  157  153  147  146  142  127  125  69  50  35  32  18  43
200  196  183  157  153  147  146  142  127  125  69  50  43  35  32  18
Binary after insertion sort :
200   196   183   157   153   147   146   142   127   125   69   50   43   35   32   18

142 -> 50 -> 125 -> 69 ->
Binary search unsuccesful!
closest node num : 11
closest number : 69
```

해당 숫자가 리스트에 없을 경우 가장 가까운 노드의 순서와 숫자를 반환한다.

```
Binary before insertion order :
10  57  184  152  151  135  128  190  153  38  117  50  124  67  71  82

enter the number that you want to search : 4
Result :
Insertion sort :
57  10  184  152  151  135  128  190  153  38  117  50  124  67  71  82
184  57  10  152  151  135  128  190  153  38  117  50  124  67  71  82
184  152  57  10  151  135  128  190  153  38  117  50  124  67  71  82
184  152  151  57  10  135  128  190  153  38  117  50  124  67  71  82
184  152  151  135  57  10  128  190  153  38  117  50  124  67  71  82
184  152  151  135  128  57  10  190  153  38  117  50  124  67  71  82
190  184  152  151  135  128  57  10  153  38  117  50  124  67  71  82
190  184  153  152  151  135  128  57  10  38  117  50  124  67  71  82
190  184  153  152  151  135  128  57  38  10  117  50  124  67  71  82
190  184  153  152  151  135  128  117  57  38  10  50  124  67  71  82
190  184  153  152  151  135  128  117  57  50  38  10  124  67  71  82
190  184  153  152  151  135  128  124  117  57  50  38  10  67  71  82
190  184  153  152  151  135  128  124  117  67  57  50  38  10  71  82
190  184  153  152  151  135  128  124  117  71  67  57  50  38  10  82
190  184  153  152  151  135  128  124  117  82  71  67  57  50  38  10
Binary after insertion sort :
190   184   153   152   151   135   128   124   117   82   71   67   57   50   38   10

124 -> 67 -> 50 -> 38 -> 10 ->
Binary search unsuccesful!
closest node num : 16
closest number : 10
```

마지막 노드의 숫자보다 작은 수를 탐색할 때

```
Binary before insertion order :
115  45  109  32  126  146  16  198  15  7  53  134  36  178  56  143

enter the number that you want to search : 199
Result :
Insertion sort :
115  109  45  32  126  146  16  198  15  7  53  134  36  178  56  143
126  115  109  45  32  146  16  198  15  7  53  134  36  178  56  143
146  126  115  109  45  32  16  198  15  7  53  134  36  178  56  143
198  146  126  115  109  45  32  16  15  7  53  134  36  178  56  143
198  146  126  115  109  53  45  32  16  15  7  134  36  178  56  143
198  146  134  126  115  109  53  45  32  16  15  7  36  178  56  143
198  146  134  126  115  109  53  45  36  32  16  15  7  178  56  143
198  178  146  134  126  115  109  53  45  36  32  16  15  7  56  143
198  178  146  134  126  115  109  56  53  45  36  32  16  15  7  143
198  178  146  143  134  126  115  109  56  53  45  36  32  16  15  7
Binary after insertion sort :
198   178   146   143   134   126   115   109   56   53   45   36   32   16   15   7

109 -> 143 -> 178 -> 198 ->
Binary search unsuccesful!
closest node num : 1
closest number : 198
```

첫번째 노드보다 큰 수를 탐색할 때

❑ 고찰

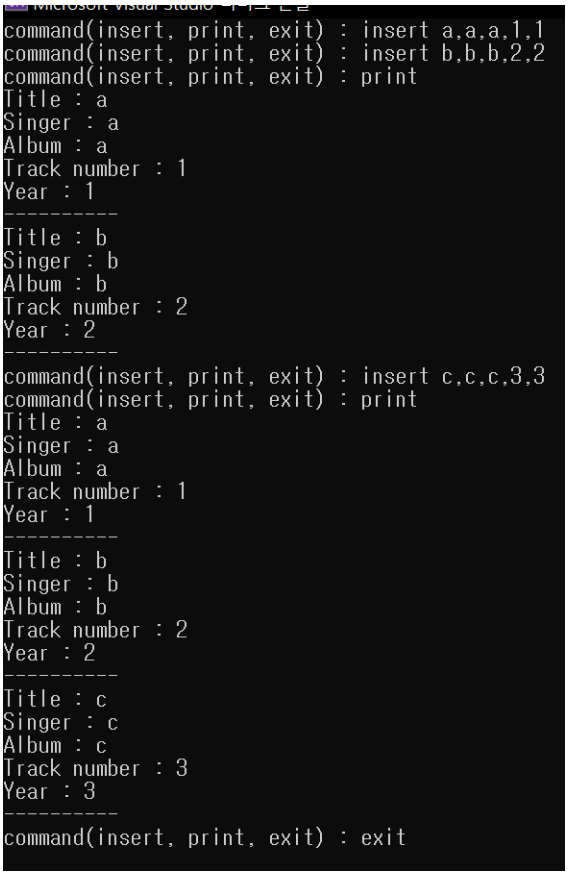
insertion sort 를 할 때 여러가지 경우로 나눠야 하는데 모든 경우를 커버하면서 효율적인 코드를 짜는 것이 어려웠다. 노드를 맨 앞에 삽입하는 경우와 중간에 삽입하는 경우, 옮기는 노드가 맨 마지막 노드일 경우 등 여러가지 경우를 생각해봐야 했다. 따라서 삽입하는 조건문 안에 삭제 조건문을 추가해 모든 경우를 고려할 수 있도록 구현했다. 또한 binary search algorithm 에서 맨 앞 노드보다 큰 숫자를 탐색할 경우와 맨 뒤 노드보다 작은 숫자를 탐색할 때 오류가 나서 따로 예외 처리를 해주었다.

Program 2

❑ 문제 설명

myMusic 클래스를 선언해 노래, 가수, 앨범명, 트랙 번호, 발매 연도를 저장하는 프로그램이다. 변수들은 모두 private 으로 선언되어 있기 때문에 public 에 선언되어 있는 함수에서 변수들에 접근해야 한다.
나 같은 경우 main 함수에서 객체 배열을 선언해 최대 100 곡의 곡을 저장할 수 있도록 구현했다. 노래를 추가하고 저장되어 있는 노래를 출력할 수 있다.

❑ 결과 화면



차례대로 정보가 저장되고 출력되는 모습을 확인할 수 있다.

❑ 고찰

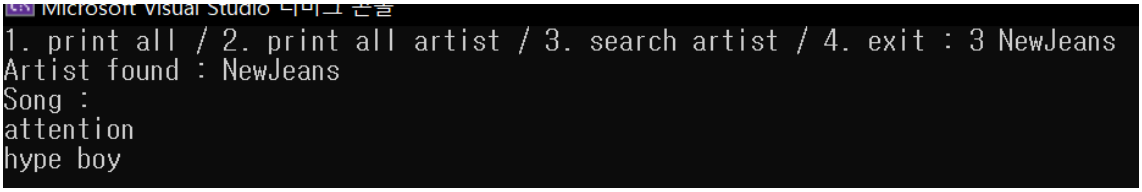
기본적인 class 를 사용한 프로그램이라 구현이 어렵지는 않았다. 다만 문제 3 번에서 사용될 기능을 부분적으로 체험할 수 있어 3 번에 대한 힌트를 얻을 수 있었던 문제였다. 다만 출력에 대한 양식이 없어서 객체 배열을 선언했던 것인데 멤버 변수로 포인터를 선언해 연결리스트로 정보를 저장한다면 개수 제한 없이 곡 정보를 저장할 수 있을 것이다.

Program 3

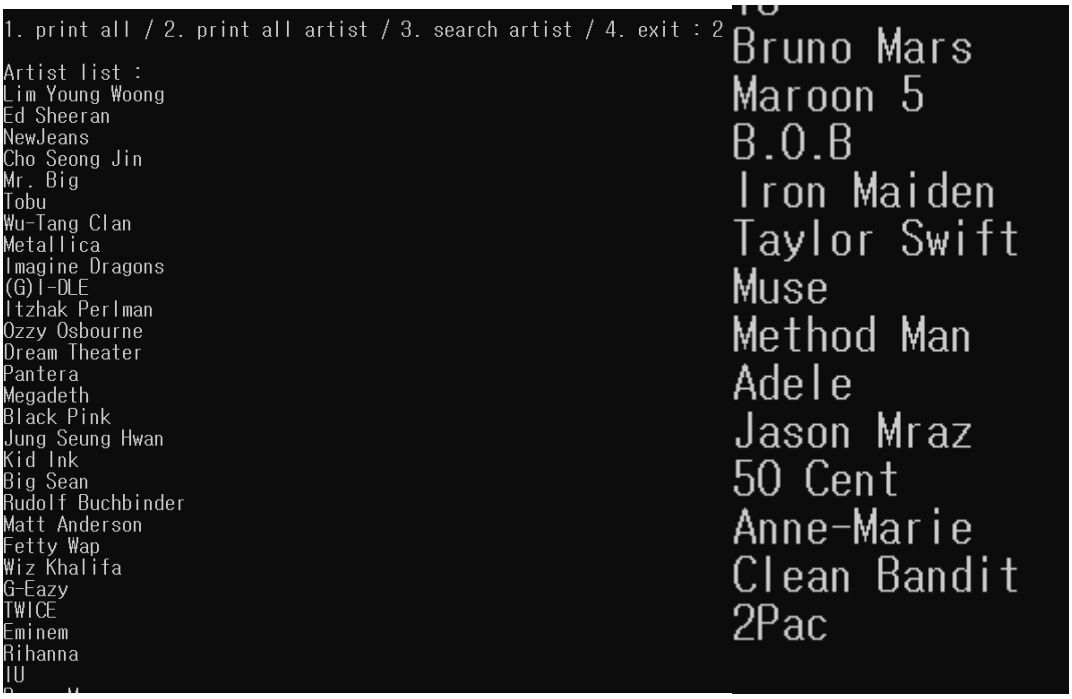
❑ 문제 설명

음악 관리 프로그램을 짜는 문제이다. 총 3 개의 클래스를 쓰는데 각각 가수 관리 class, 가수 정보 class, 곡 정보 class 이다. 가수 관리 클래스에서는 가수 정보들을 연결 리스트로 연결하는데 선형 연결 리스트로 마지막 노드가 첫 노드를 가리키면서 리스트가 원형으로 순환한다. 각 가수 노드에서 곡 노드를 가리켜 해당 가수의 곡 정보를 연결 리스트로 저장한다. 가수와 곡 정보는 텍스트 파일에 저장되어 있어 파일 입출력을 사용해 정보를 추출, 저장한다.

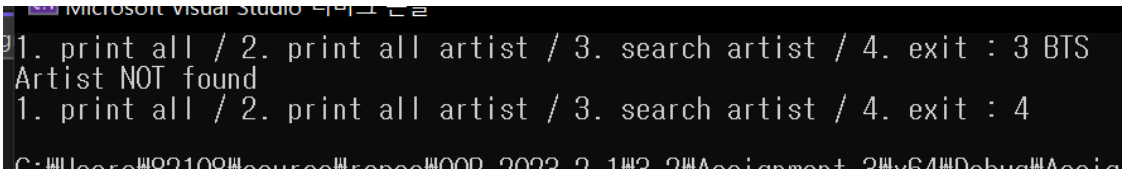
❑ 결과 화면



가수 NewJeans 를 찾아 해당 가수의 곡까지 출력한다.



가수들만 출력한다.



음악 관리 프로그램에 존재하지 않는 가수 검색

❑ 고찰

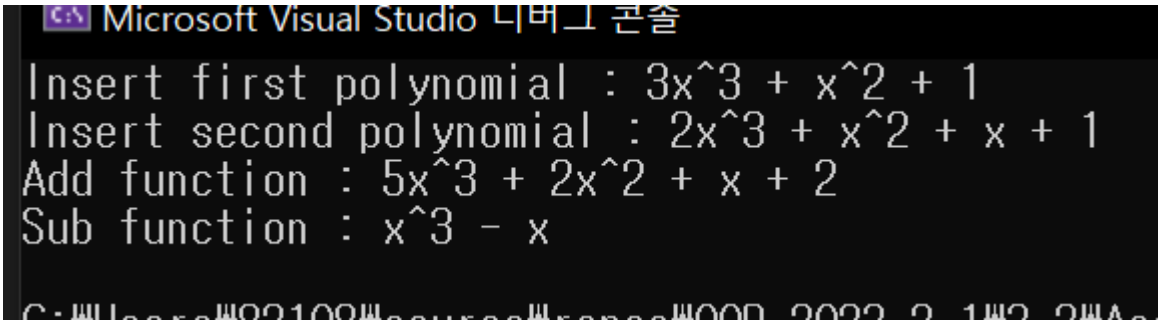
원형 연결 리스트는 마지막 노드가 첫 노드를 가리키기 때문에 마지막 노드라는 개념이 성립하지 않는다. 따라서 처음으로 삽입한 노드를 head 라고 했을 때 head->getprev()를 했을 때 반환되는 노드를 마지막 노드라고 설정했다. 원형 연결 리스트를 출력할 때나 메모리 할당해제를 할 땐 bool 변수를 선언하고 curNode 를 움직이다가 만약 curNode 가 다시 head 를 가리키면 bool 변수를 변화시켜 출력을 종료하도록 구현했다. 또한 곡 이름은 다 소문자로 변환한 후 노드를 삽입하기 때문에 모든 출력에서 소문자로 출력된다.

Program 4

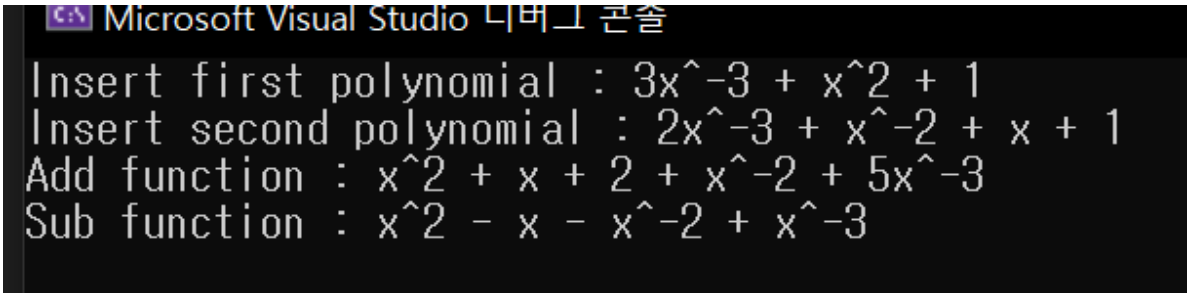
❑ 문제 설명

이차 방정식의 덧셈과 뺄셈을 구하는 프로그램이다. 두 이차식을 입력 받아 합, 차를 연산한 후 결과값을 출력해준다. 지수는 음수가 될 수 있으며 미지수는 x 로 한정되어 있지 않다. 각 이차방정식에서 동일한 차수가 2 번 이상 입력될 수 없으며 결과값은 차수의 내림차순으로 저장되어 있어야 한다. 각 Term 은 계수와 차수를 저장하고 있으며 class Polynomial 에 의해 연결되어 있다.

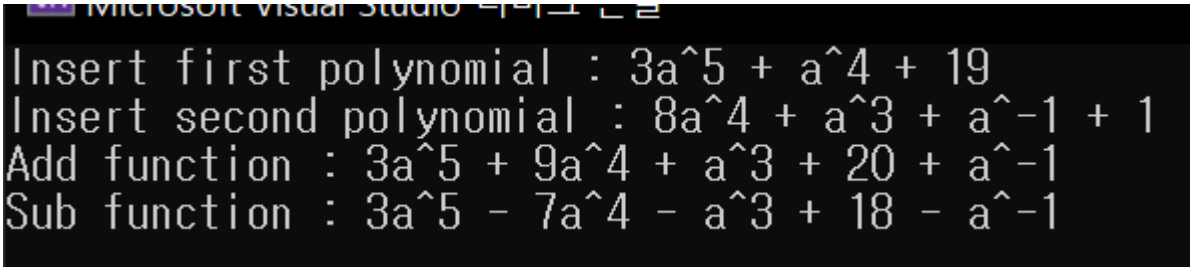
❑ 결과 화면



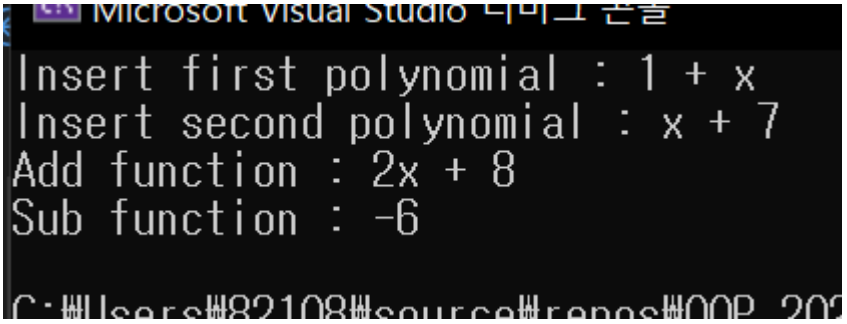
문제지 예시. 2 개의 이차방정식의 합, 차를 출력



지수가 음수일 때



미지수가 x 가 아닐 때



상수가 마지막에 위치하지 않을 때

❑ 고찰

프로그램을 짤 때 고려해야 할 예외사항이 너무 많았다. 계수와 차수가 1 일 땐 숫자가 생략되고 차수가 0 일 땐 미지수가 표현되지 않으며 지수가 음수일 경우도 고려해야 했다. 입력을 받을 때 부호와 차수의 양, 음을 구분하는 등 생길 수 있는 예외가 너무 많아 모든 경우를 만족하는 함수를 만드는 것이 힘들었다. 또한 연산할 때 계수가 0 이 되면 결과값에 저장하지 않는 것과 두 이차식에서 공통되지 않는 차수의 연산까지 고려하는 것이 너무 어려웠다.

따라서 노드를 삽입할 때 bool 변수를 적극적으로 활용해 계수가 저장된 상태인지, 숫자가 존재하는지, 1 이 생략이 된 것인지 아닌지 등을 확인했다. 출력할 땐 계수의 부호에 따라 "+","-"를 출력하고 값을 양수로 바꿔주는 형식으로 예외를 처리했다.