객체지향프로그래밍 2차 과제 보고서

2021202085 전한아솔

먼저, string 관련 함수를 사용하지 못하기 때문에, 자주 사용하는 strlen, strcpy, strcmp함수를 직접 구현했다.
먼저 strlen은 문자열을 인자로 전달받고 int len=0으로 초기화했다. for 문을 통해 I가 0부터 널 문자를 만나기 전까지 반복할 때마다 len 변수를 증가시켜서 길이를 반환했다.
strcpy는 문자열 2개를 인자로 전달받아, while문을 사용하여 복사할 문자열의 끝까지 반복하며 나머지 한 문자열에 문자를 하나씩 복사한다. 이후 문자열에 끝에 널 문자를 붙여서 문자열을 완성한다.
strcmp는 문자열 2개를 인자로 전달받아서 while문을 통해, 한 문자열의 문자가 NULL이 아니고 I 인덱스의 문자가 같다면 I를 증가시킨다. 반복이 끝나고 두 문자열의 I 인덱스의 값이 같다면 0을 반환, 앞 문자열이 뒷 문자열보다 I 인덱스의 값이 크다면 1, 작다면 -1을 반환하도록 구현했다.

<Assignment #2-1>

1. Write a program that dynamically creates an array of random size (between 5 and 20), fills each of its elements with random integers ranging from 0 to 100, and then prints all the values in the array, as well as the maximum and minimum values and the addresses of those maximum and minimum values. If there are two or more maximum or minimum values, print the address of the value with the lowest array index.

```
Ex)
Size of the array : 6
Random numbers : 86 87 82 62 46 14

Maximum value: 87          , Address: 000002922BA026D4
Minimum value: 14          , Address: 000002922BA026E4
```

5~20사이즈의 랜덤한 값을 입력받아서 그에 맞는 크기의 배열을 만들고 랜덤한 값을 할당한 후 최댓값과 그 최댓값이 저장된 주소, 최솟값과 그 최솟값이 저장된 주소를 출력한다. 이때, 최대·최솟값이 중복된다면 가장 앞의 인덱스의 주소를 출력한다.
<알고리즘>
먼저, 실행마다 랜덤한 값을 받기 위해 srand의 시드를 time(NULL)로 저장하고 나머지 연산자를 이용하여 랜덤 값의 범위를 제한한다. 문제의 범위는 5~20이므로 %16을 통해서 0~15의 값이 나오게 하고 +5를 통해서 이 범위를 5~20으로 제한하도록 했다. 이렇게 저장한 변수(size)만큼의 크기를 동적 할당하여 int형 포인터를 생성하고 rand()%101을 통해서 0~100의 랜덤 값을 0~size-1인덱스에 저장한다.
최대·최솟값을 출력하기 위해서 O(n^2)의 시간복잡도를 가지지만 구현이 가장 쉬운 bubble

sort를 사용하여 배열을 오름차순으로 정렬했다. 오름차순을 버블소트로 구현하는 알고리즘은 현재 인덱스의 값과 뒤 인덱스의 값을 비교하여 앞의 값이 뒤보다 크다면 바꾸어주는 식이다. 이렇게 1번 실행하면 맨 뒤는 가장 큰 값이 저장되므로 다음에는 맨 뒤보다 하나 앞 인덱스까지만 비교하면 된다.
이렇게 정렬한 후, 가장 앞은 최솟값, 가장 뒤는 최댓값이 저장되어 있으므로 arr[0]와 arr[max_point](max_point=size-1로 초기화 해놓음)을 출력하면 된다. 하지만 최솟값은 중복되어도 arr[0]이 가장 앞 인덱스이므로 그냥 출력해도 되지만 최댓값은 중복된다면 max_point의 인덱스가 가장 빠른 최댓값이 아니다. 따라서 배열의 맨 뒤부터 앞까지 반복하면서 arr[i]와 arr[i-1]의 값이 같다면 max_point를 하나 줄여주었다. 배열의 맨 뒤는 최댓값이고, 앞 인덱스가 그 값과 같다는 것은 최댓값이 중복된다는 것이고, 값이 다르다면 최댓값이 아니라는 뜻이므로 반복문을 통해서 최댓값이 저장된 가장 앞 인덱스까지 max_point를 감소시켜 출력했다.
<결과>

```
Size of array: 16
Randon numbers: 16 37 29 67 21 14 63 4 18 55 82 4 34 97 15 23

Maximum value: 97     ,Address: 00000198A757A19C
Minimum value: 4      ,Address: 00000198A757A160
```

랜덤한 16개의 배열 중 최솟값은 4, 최댓값은 97이므로 각 인덱스의 주소를 출력한다. 이때 최솟값인 4가 겹치는데, 오름차순으로 정렬했기 때문에 가장 앞 인덱스인 arr[0]을 이용하여 최솟값과 주소를 출력했다.

<Assignment #2-2>

0~100의 10가지 랜덤 정수를 퀵 정렬, 병합 정렬 두 가지 방법 중 한 방법을 선택하여 오름차순으로 정렬하고 이진 탐색을 통해서 찾는 값이 있다면 값의 인덱스를 출력, 값이 없다면 오름차순에 맞게 값을 넣어서 출력하는 프로그램이다.
찾는 값이 없다면 그 값을 넣어서 출력해야 하므로 배열의 크기를 11로 저장했다. 1번과 같은 방식으로 %101을 통해서 0부터 100의 랜덤 값을 저장한다.

2. Write a program that generates 10 random integers(0 to 100) and sorts the values in ascending order using **quick sort** and **merge sort**. Then implement a binary search algorithm to determine if the given integer exists (if not, insert it in the sorted position). Include in the report an analysis of sorting algorithms based on their time complexity.

```
Ex1)
Random values :  74 58 39 84 42 24 49 50 74 52
Select sorting method (1: Quick Sort, 2: Merge Sort) : 1
Sorted numbers (Quick Sort): 24 39 42 49 50 52 58 74 74 84
Enter a value to search : 50
Searched number index : 4
```

```
Ex2)
Random values :  74 58 39 84 42 24 49 50 74 52
Select sorting method (1: Quick Sort, 2: Merge Sort) : 1
Sorted numbers (Quick Sort): 24 39 42 49 50 52 58 74 74 84
Enter a value to search : 51
Updated numbers: 24 39 42 49 50 51 52 58 74 74 84
```

<알고리즘>
먼저 퀵 정렬은 가장 맨 앞의 값을 key로 저장한 후, i는 다음 인덱스부터 key보다 큰 값을 만나면 멈추고 j는 맨 뒤 인덱스부터 key보다 작은 값이 나오면 멈추어 두 수를 바꾼다. 이렇게 반복하다가 i와 j가 엇갈린다면 j와 key의 값을 바꾼다. 이러면 j의 왼쪽 배열에는 key보다 작은 값, j의 오른쪽 배열에는 key보다 큰 값이 저장될 것이다. 따라서 이 j를 기준으로 왼쪽, 오른쪽으로 분할하고 다시 같은 알고리즘을 적용하여 반복하면 된다.
이는 분할 정복 알고리즘을 사용하는 정렬로, 분할하는 과정에서 한 수를 정하고 그 수보다 작으면 왼쪽, 크면 오른쪽에 저장하는 '파티셔닝'을 사용하는데, 이 파티셔닝은 모든 수를 비교해야 하므로 시간복잡도가 O(n)이다. 배열의 요소가 무한히 커지면 평균적으로 가운데의 수를 골라서 수행하므로, 수행마다 배열이 절반으로 줄어들어서 O(log n)의 시간복잡도를 가진다. 따라서 퀵 정렬의 시간복잡도는 O(n·log n)이다. 하지만 최댓값이나 최솟값을 key로 설정한 최악의 경우는 모든 수를 비교해야 하므로 시간복잡도는 O(n^2)이다.

병합 정렬은 배열의 중간을 기준으로 배열을 두 개로 나눈다. 또 이 두 개의 배열을 중간을 기준으로 나누는 것을 반복하여 배열이 1개가 될 때까지 반복한다. 이후 분할된 배열을 거꾸로 올라가서 1개의 배열과 1개의 배열을 비교하여 오름차순으로 정렬해야 하므로 작은 값을 먼저 저장, 나머지를 저장하여 2개의 배열로 병합한다. 이런 방식으로 2개의 배열끼리 비교하여 4개의 배열로 병, 4개의 배열끼리 비교하여 병합을 반복하는 알고리즘을 사용한다. 분할하는 함수와 병합하는 함수를 나누어서 구현했다.

먼저 분할된 배열을 정렬하여 병합하는 함수(merge)는 첫 인덱스와 마지막 인덱스의 중간 값을 저장하는 int mid를 선언했다. int i를 선언하여 I에는 첫 인덱스를 저장, int j를 선언하여 j에는 mid+1을 선언하여 매개변수로 전달된 배열인 arr의 앞, 뒤 배열을 탐색하는 용도로 사용한다. I는 mid보다 작고 j는 end보다 작을 때까지 반복한다. (각 배열의 끝에 도달하기 전까지 반복한다). 이런 반복에서 arr[i]와 arr[j]를 비교하여 더 작은 값을 새로운 배열인 temp에 저장한다. 이런 식으로 I와 j를 증가시키면서 값을 비교하다가 I가 mid보다 커진다면 temp의 앞 배열(0~mid 인덱스)이 모두 채워졌고, 뒷 배열(mid~end)은 이미 정렬되어 있으므로 arr[j]를 temp 저장하면 된다. j가 end보다 커질 때도 마찬가지로 arr[i]를 temp에 저장하여 남은 값들을 넣어준다.

분할하는 함수(divide)는 배열의 첫 인덱스와 마지막 인덱스를 받아서 중간값을 저장하고 처음부터 중간, 중간부터 끝의 범위로 나누어서 다시 divide 함수에 첫 인덱스와 마지막 인덱스로 전달하여 호출하는 재귀의 방식을 사용했다. 처음부터 중간까지 범위를 나누는 divide를 호출하고 중간부터 끝까지 범위를 나누는 divide를 호출한 후 merge를 호출했다. 이런식으로 함수를 구성하면 배열의 분할은 재귀적으로 호출하므로 첫 배열의 왼쪽 (처음부터 중간)을 배열이 한 개가 될 때까지 나누고, 가장 왼쪽 두 배열을 병합, 그 다음 두 배열을 병합..을 반복한다. 첫 배열의 오른쪽(중간부터 끝)도 똑같이 배열이 한 개가 될 때까지 나누고 왼쪽부터 병합을 반복, 이렇게 병합하여 만든 배열 두 개를 병합하여 최종적으로 정렬된 배열을 구성하게 된다.

이렇게 정렬한 배열을 이진 탐색으로 찾아야한다. 이진 탐색은 이미 정렬된 배열의 중간값을

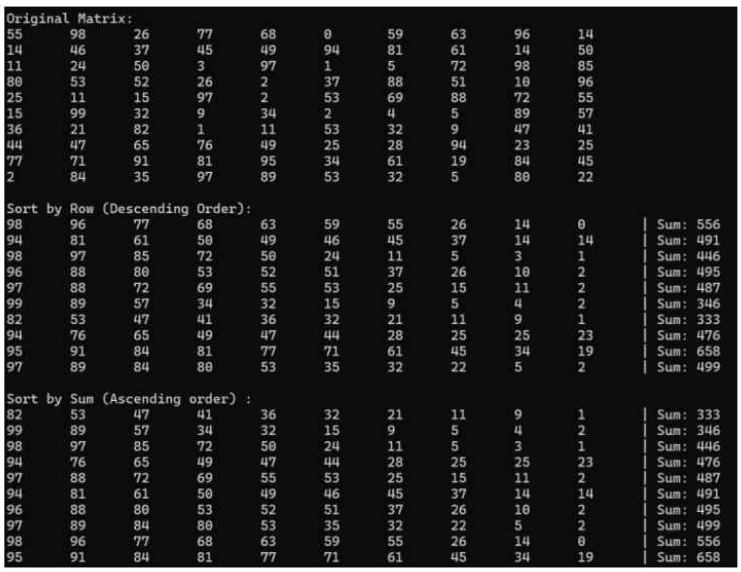
기준으로 내가 찾을 값이 더 작은지(오름차순을 기준으로 왼쪽에 있는지), 더 큰지(오른쪽에 있는지) 비교하여 중간값의 위치를 바꾸어서 찾는다. 이렇게 찾게되면 한 번의 실행마다 찾을 배열이 1/2이 되므로 시간복잡도는 O(log n)이 된다. start=0, end=배열의 길이-1로 초기화하고 start가 end보다 작거나 같을 때 까지 반복한다. int mid=(start+ end)/2로 설정하여 mid와 찾는 값이 같다면 true를 반환한다. 내가 찾는 값이 mid의 값보다 작다면 end를 mid-1로 초기화하여 왼쪽 부분의 배열만 찾도록 하고, mid보다 크다면 start를 mid+1로 초기화하여 오른쪽 부분의 배열만 찾도록 한다. 이 때 함수의 반환형은 bool형으로 값을 찾으면 true, 반복문을 벗어날 때까지 값을 못 찾으면 false를 반환한다. 함수의 인자로 int& index를 선언했는데, 값을 찾지 못했을 때, index를 start로 초기화한다. 이렇게 되면 I와 I+1의 인덱스 사이에 내가 찾는 값이 있을 때, start는 I가 되므로 start를 저장하여 index를 통해서 내가 찾을 값이 없다면 배열에 오름차순으로 값을 끼워넣을 수 있다.
<결과>

```
Random values : 40 48 74 36 38 48 49 3 48 0
Select sorting method (1: Quick Sort, 2: Merge Sort) :2
Sorted numbers (Merge Sort): 0 3 36 38 40 48 48 48 49 74
Enter a value to search : 36
Searched number index : 2
Random values : 51 53 58 100 82 61 95 93 43 98
Select sorting method (1: Quick Sort, 2: Merge Sort) :1
Sorted numbers (Quick Sort): 43 51 53 58 61 82 93 95 98 100
Enter a value to search : 52
Updated numbers: 43 51 52 53 58 61 82 93 95 98 100
```

첫 번째 결과에서는 병합 정렬을 사용하여 수를 정렬하고 찾고 싶은 값인 36을 입력하자 36의 인덱스를 반환한다. 두 번째 결과는 퀵 정렬을 사용하여 수를 정렬하는데, 찾고 싶은 값인 52가 없으므로 정렬된 배열의 오름차순에 맞게 수를 삽입하여 배열을 만들었다.
<고찰>
항상 버블정렬을 사용하여 배열을 정렬했는데 2번 문제의 퀵 정렬과 병합 정렬을 배우고 이번 프로그램을 만들어 보면서 처음엔 특히 정렬하는 과정이나 순서가 이해가 잘 가지 않아서 여러 방식을 살펴보며 이해했다. 시간복잡도를 설명하기 위해 찾아보는 과정에서 처음엔 버블정렬과 시간복잡도가 비슷한 줄 알았으나 배열이 커지면 커질수록 엄청난 차이를 보인다는 것도 깨달았다.

<Assignment #2-3>

3. Write a program that prints a 10x10 matrix filled with random integers(0 to 100), stores the matrix in a variable of type int**, sorts and reprints the matrix by the **descending order** of the rows, and sorts and reprints the matrix by the **ascending order** of the sum of the rows. When sorting by the sum of the rows, you should replace the address pointed to by the pointer, not replace the value directly. Print the values with tab escape sequences between numbers (Please refer to the example below)



10×10의 2차원 배열에 0~100의 랜덤 값을 할당 후 해당 열을 내림차순으로 정렬하고 합 값을 출력하고 각 행의 합 값으로 오름차순 정렬을 실행하여 합 값에 맞는 행을 바꾸어서 출력하는 프로그램이다. 10×10 매트릭스는 int**형으로 선언한다.

<알고리즘>

매트릭스가 int**형이므로 int*의 크기를 10만큼 동적 할당하여 행의 메모리를 만들고 for 문을 통해서 0~9까지 각 행의 열을 int의 크기로 10만큼 동적 할당하여 10×10 매트릭스의 메모리를 만들었다. 또한 각 행의 합 값을 오름차순으로 정렬하여 행을 바꿔줘야 하므로 각 행의 첫 번째 인덱스의 주솟값이자, 행을 저장할 포인터 배열도 크기 10으로 선언하였고, 각 행의 합 값을 저장할 int형 배열도 크기 10으로 선언과 동시에 배열의 값을 0으로 초기화했다.

2중 for 문을 통해서 각 행과 열에 랜덤 값을 부여했는데, 이는 1, 2번에서 사용했듯 0~100의 숫자 중 하나이므로 rand() % 101을 통해서 값을 저장했다. 또한 각 행의 합 값도 저장해야 하므로 랜덤 값을 저장하면서 합을 저장하는 배열에 더해주었다. 이렇게 저장된 행의 값을 내림차순으로 정렬하기 위해서 1번에서 사용했던 버블정렬 알고리즘을 사용하여 정렬했는데, 이때 내림차순으로 정렬해야 하므로 앞 인덱스의 값이 뒤 인덱스의 값보다 작으면 두 값을 바꾸는 방식으로 정렬했다. 이렇게 내림차순으로 정렬해도 각 행의 합은 바뀌지 않으므로 내림차순으로 정렬한 행을 출력하고 합 값은 그대로 출력했다.

각 행의 첫 번째 인덱스의 주솟값을 포인터 배열에 저장하고 이 포인터 배열을 통해서 합 값을 오름차순으로 정렬해야 한다. 따라서 각 행을 인자로 전달받아 행의 합 값을 반환하는 함수를 만들었다. 포인터 배열에는 각 행의 첫 번째 인덱스의 주소가 들어있으므로 포인터 배열을 통해서 매트릭스에 접근할 수 있고, 이를 통해서 매트릭스의 행을 전달하여 버블정렬의 알고리즘으로 행을 바꾸었다. 이때 한 행의 합 값이 다음 행의 합보다 크다면 두 행을 바꾸어 준다. 이렇게 되면 포인터 배열에 들어있는 행이 오름차순으로 정렬되고 포인터 배열은 2차원 배열처럼 arr[i][j]의 방식을 사용하여 접근할 수 있으므로 포인터 배열을 사용하여 오름차순으로 정렬된 매트릭스를 출력했다.

<결과>

```
Original Matrix
67      47      16      75      73      82      20      56      94      63
70      78      79      86      52      5       72      90      9       90
48      60      44      77      89      97      12      45      95      92
65      93      18      68      79      9       82      63      77      37
76      41      57      62      80      98      25      52      74      38
64      41      12      40      2       86      16      54      7       14
50      19      47      99      41      33      63      4       85      29
46      41      90      99      25      3       92      30      70      93
90      46      66      88      26      33      37      39      70      89
65      29      8       86      96      13      21      64      66      59

Sort by Row (Descending Order):
94      82      75      73      67      63      56      47      20      16      | Sum: 593
90      90      86      79      78      72      70      52      9       5       | Sum: 631
97      95      92      89      77      60      48      45      44      12      | Sum: 659
93      82      79      77      68      65      63      37      18      9       | Sum: 591
98      80      76      74      62      57      52      41      38      25      | Sum: 603
86      64      54      41      40      16      14      12      7       2       | Sum: 336
99      85      63      50      47      41      33      29      19      4       | Sum: 470
99      93      92      90      70      46      41      30      25      3       | Sum: 589
90      89      88      70      66      46      39      37      33      26      | Sum: 584
96      86      66      65      64      59      29      21      13      8       | Sum: 507

Sort by Sum (Ascending order):
86      64      54      41      40      16      14      12      7       2       | Sum: 336
99      85      63      50      47      41      33      29      19      4       | Sum: 470
96      86      66      65      64      59      29      21      13      8       | Sum: 507
90      89      88      70      66      46      39      37      33      26      | Sum: 584
99      93      92      90      70      46      41      30      25      3       | Sum: 589
93      82      79      77      68      65      63      37      18      9       | Sum: 591
94      82      75      73      67      63      56      47      20      16      | Sum: 593
98      80      76      74      62      57      52      41      38      25      | Sum: 603
90      90      86      79      78      72      70      52      9       5       | Sum: 631
97      95      92      89      77      60      48      45      44      12      | Sum: 659
```

첫 매트릭스의 각 행을 내림차순으로 정렬하여 각 행의 합 값까지 출력한다. 이 합 값을 가지고 오름차순으로 행을 바꾸어서 정렬된 매트릭스가 출력되고 각 행의 내림차순은 그대로 유지된다.

<고찰>

합 값을 가지고 어떻게 행 전체를 바꿀지 생각하던 중 포인터 배열을 떠올렸다. 포인터 배열은 각 배열의 요소를 포인터로 가지므로 행의 주소를 요소로 가지도록 하여 행의 합 값을 통해 행의 주소를 바꾸고 포인터 배열을 통해서 출력하면 바뀐 행으로 출력되도록 구현했다. 하지만 처음에는 포인터 배열의 선언이나 초기화를 어떻게 해야할지 너무 헷갈려서 여러 차례 이상한 코드를 구성했다. 하지만 이번 문제를 통해서 포인터 배열을 헷갈리지 않고 사용할 수 있게 되었다.

<Assignment #2-4>

4. Write a program to separate strings by the given delimiters. Delimiters can be words, including spaces. (The use of the string library is prohibited)
   - The input string can be up to 100 characters
   - The input delimiter can be up to 10 characters

```
Ex)
Enter the string : C:₩Users₩Desktop₩2024₩OOP₩Assignment
Enter the delimiter : ₩

Separated tokens :
C:
User
Desktop
2024
OOP
Assignment
```

문자열과 구분자를 입력받아서 구분자로 문자열을 구분하여 출력하는 프로그램이다. 구분자는 단어나 공백도 포함할 수 있다.

<알고리즘>

문자열이나 구분자는 공백을 포함할 수 있으므로 getline을 사용하여 공백까지 문자열에 포함하도록 입력받았고 구분자로 구분한 문자열을 저장할 2차원 배열인 temp를 선언했다. 문자가 구분자와 같은지 판단해주는 bool형 함수인 Check 함수를 선언했는데, 구분자로 분리되는 단어가 한 문자가 아니거나 구분자 또한 단어가 될 수 있으므로 Check 함수의 인자로 char*형을 선언하여 for 문을 통해서 I 인덱스의 문자가 같지 않으면 true, 같다면 flase를 반환하도록 정의했다. mian 함수에서는 while 문을 사용하여 문자열의 끝에 도달하기 전까지 반복하는데, Check함수에 문자열의 I 인덱스의 주솟값을 인자로 전달하여 문자열의 한 문자씩 비교할 수 있도록 했다. Check가 true이면 구분자와 문자열이 다르므로 temp의 한 행에 문자열을 저장하고 false라면 행의 끝에 널 문자를 삽입하여 한 행을 끝내고 다음 행으로 넘어간다. 이때 다음 행의 처음부터 시작할 수 있도록 열을 순회하는 변수를 0으로 초기화하고 문자열을 순회하는 I는 구분자의 길이-1 만큼을 더해주어서 구분자를 건너뛰고 다음 문자부터 저장하도록 구현했다.

<결과>

```
Enter the string : a#b#c#d#f
Enter the delimeter : #
Separated tokens :
a
b
c
d
f
```

a#b#c#d#f를 입력하여 #를 기준으로 문자를 나누어서 출력한다.

<고찰>

구분자가 단어도 가능하므로 첫 구분자를 만났을 때 나머지를 쭉 비교하고 단어의 길이-1을 더하여 구분자를 넘어가서 다음 문자부터 비교하는 방식으로 구현했는데, 처음에는 구분자가 단어라면 어떻게 단어끼리 비교하고 구분자를 넘어가야 하는지 생각이 나지 않아 애먹었던 문제라고 생각한다.

<Assignment #2-5>

5. A Hadamard matrix is a special form of a binary matrix in which every row and column has a different bit pattern. Write a program to generate a Hadamard matrix that satisfies the following conditions

   Step 1 : Accept an input n from the user. ('n' represents a power of 2)

   Step 2 : Generate a $2^n \times 2^n$ Hadamard matrix

   Step 3 : Display the generated Hadamard Matrix.

   Hadamard matrix is generated in a recursive way. Solve the problem using the method of using a smaller matrix to generate a larger matrix. The example below generates a Hadamard matrix of size 8x8

$$H_{2^0} = [1],\ H_{2^1} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix},\ H_{2^2} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}, H_{2^n} = \begin{bmatrix} H_{2^{n-1}} & H_{2^{n-1}} \\ H_{2^{n-1}} & -H_{2^{n-1}} \end{bmatrix}$$

```
Enter the value of n for Hadamard matrix (2^n x 2^n): 3

Hadamard Matrix of size 8x8:
1       1       1       1       1       1       1       1
1       -1      1       -1      1       -1      1       -1
1       1       -1      -1      1       1       -1      -1
1       -1      -1      1       1       -1      -1      1
1       1       1       1       -1      -1      -1      -1
1       -1      1       -1      -1      1       -1      1
1       1       -1      -1      -1      -1      1       1
1       -1      -1      1       -1      1       1       -1
```

n을 입력하면 2^n × 2^n의 크기에 아다마르 행렬을 생성하는데, 각 아다마르 행렬의 마지막 2^(n-1) × 2^(n-1) 크기의 행렬은 음수를 붙여서 행렬을 생성하고 출력하는 프로그램이다.

<알고리즘>

n을 입력하면 2^n의 크기를 생성하기 위해서 재귀의 방법으로 2^n을 반환하는 함수를 정의했다. 이 함수를 이용하여 2^n을 size 변수에 저장하고 아다마르 행렬에 인자로 전달했다.

아다마르 행렬을 구성하는 함수의 인자는 행렬의 행과 열을 저장하기 위해 int**형 arr과 크기를 전달하는 int n, 행과 열을 나타내는 int I, j로 구성했다. 이 때 I와 n이 1이라면 행렬은 1×1의 크기를 가지고 이는 1이므로 arr[i][j]에 1을 저장하고 함수를 종료한다. n이 1이 아니라면 temp 변수에 n으로 전달된 크기의 절반을 저장하고 temp를 다시 함수에

전달하여 재귀적으로 행렬을 구성한다. 1구역을 왼쪽 위, 2구역을 오른쪽 위, 3구역을 왼쪽 아래, 4구역을 오른쪽 아래로 설정하겠다. 아다마르 행렬이 4×4의 크기라고 가정하고 설명하면, temp에는 2가 저장되고 모든 구역은 2×2로 쪼개어 4구역으로 나뉘어서 구성될 것이다. 2×2 크기의 1구역을 구성하려면 또 1×1의 크기의 4구역으로 나뉘어 구성하는데, 2×2 크기의 행렬에서 temp=1이고, temp를 다시 함수로 전달하여 호출할 때 n=temp=1이므로 arr[0][0]=1이 된다. 따라서 1×1의 1구역이 구성된다. 이제 1×1의 2구역을 구성하기 위해서 함수의 크기는 똑같이 temp로 전달하지만 2구역은 오른쪽에 윗부분이므로 j 대신 j+temp를 전달한다. temp=1이므로 I=0, j=1이 되고 n=temp=1이므로 arr[0][1]=1이 된다. 3구역은 왼쪽 아랫부분이므로 I 대신 I+temp를 전달하여 I=1, j=0이 되고, arr[1][0]=1이 된다. 이제 4구역을 구성해야 하는데, 4구역은 나머지 구역과 반대 부호를 가지도록 구성한다. 따라서 2중 for 문을 이용하여 temp만큼 반복하도록 하고 3구역의 인덱스와 모두 반대로 구성하기 위해서 -를 붙여주면 4구역은 3구역과 반대 부호를 가지므로 1×1 크기의 4구역은 -1이 된다. 따라서 2×2 크기의 1구역은 1×1 크기의 구역 순서대로 1, 1, 1, -1로 구성된다. 2×2 크기의 2, 3구역 또한 같은 방식으로 구성되고, 4×4 크기의 행렬에서 temp는 2이므로 2×2 크기의 4구역도 마찬가지로 for 문을 이용해 temp만큼 반복하여 3구역의 부호와 반대로 저장하면 된다. 이런 식으로 재귀를 이용하여 4×4 -> 2×2 —> 1×1의 순서로 큰 행렬을 작은 행렬로 쪼개서 각 구역을 구성하여 아다마르 행렬을 만들도록 구현했다.

<결과>

```
Enter the value of n for Hadamard matrix (2^n x 2^n): 3
1       1       1       1       1       1       1       1
1       -1      1       -1      1       -1      1       -1
1       1       -1      -1      1       1       -1      -1
1       -1      -1      1       1       -1      -1      1
1       1       1       1       -1      -1      -1      -1
1       -1      1       -1      -1      1       -1      1
1       1       -1      -1      -1      -1      1       1
1       -1      -1      1       -1      1       1       -1
```

n이 3이라면 2^3=8이므로 8×8 크기의 아다마르 행렬이 생성된다. 4구역은 나머지 구역들의 부호와 반대임을 볼 수 있다.

<고찰>

이번 문제를 통해서 아마다르 행렬을 처음 보았는데, 재귀를 어떤 방식으로 사용하여 구현할지 많이 고민했다. 문제에서 n=0이면 행렬은 1인 초기 값을 이용하여 큰 행렬을 작게 잘라서 각 구역을 나눠 구현하며 재귀의 호출 방식, 순서에 대해서 자세히 공부하고 알 수 있었다.

<Assignment #2-6>

파일로부터 문장을 읽어와서 open이면 파일의 이름을 입력하여 해당 파일을 열고, search라면 단어를 입력받아 해당 단어의 행과 열을 출력하고, change라면 바꿀 단어와 바뀔 단어를 입력하여 해당 단어를 바꾸고, insert라면 행과 열, 단어를 입력받아 입력받은 행과 열 위치에 해당 단어를 삽입하고, delete라면 단어를 입력받아서 해당 단어를 삭제하고, save라면 저장할 파일의 이름을 입력받아 입력받은 이름으로 변경된 문장을 파일에 저장하고,

6. Write a program that takes a text file as input and modifies it based on the implemented commands. The possible commands to implement are open, search, change, insert, delete, save, and exit.

Command	Format	Description
open	open [file path]	Open the file you want to edit. Exception handling should be performed in case the file does not exist.
search	search [word]	Print the starting column and row of all words found. Expressed as (row, column)
change	change [word1] [word2]	Change all of word1 to word2
insert	insert [row] [column] [word]	Add the words you want to add to the corresponding column and row positions
delete	delete [word]	Delete all 'word'
save	save [title]	Save the file with the title you entered
exit	exit	Exit the program

Input	Output
open ./testcase_6.txt search CHAPTER search glass change Alice Lucy insert 1 16 * delete CHAPTER save ./changedtext.txt exit	===‘CHAPTER’ search(1)=== (10, 28) --- ===‘glass’ search(3)=== (126, 6), (200, 26), (217, 32) --- ===change=== Alice -> Lucy --- ===insert=== * Inserted into (1, 16) --- ===delete=== Delete CHAPTER --- ===save=== Save the file as “./changedtext.txt” --- Exit the program

exit라면 프로그램을 종료한다.

<알고리즘>

파일로부터 문장을 읽어오기 위해서 ifstream 객체를 선언하고 파일로부터 읽어온 문장을 저장하기 위해서 char형 2차원 배열을 선언했다.

<open>

열고 싶은 파일의 이름을 입력받아서 char형 배열에 저장 후 ifstream의 open 함수를 사용하여 파일을 열었다. ifstream의 getline을 사용하여 한 문장씩 받아온 후, 직접 정의한 strcpy 함수를 통해서 해당 문장을 2차원 배열의 한 행에 저장했다. 이렇게 저장할 때마다 int형 변수인 row를 증가시켜 해당 파일이 총 몇 행인지 알 수 있도록 했다.

<search>

함수의 인자로는 2차원 배열의 한 행(char*), 내가 찾을 단어(char*), 단어를 찾았는지 아닌지를 알 수 있는 flag는 int형 참조자로 선언하여 search를 호출하는 부분에서 flag의 값을 공유할 수 있도록 했고, 단어가 존재하는 열의 첫 인덱스를 저장하는 col도 int형 참조자로 선언하여 col의 값도 공유할 수 있도록 정의했다.

한 문장의 처음부터 널 문자를 만날 때까지 반복하여 한 문자씩 비교한다. 만약 문자가 공백이라면 열을 하나 증가시키고 continue를 이용해 다음 문자로 넘어간다. 공백이 아니라면 해당 문자부터 다음 공백을 만나기 전까지, 널 문자를 만나기 전까지, 내가 찾을 단어의 길이만큼 반복하며 해당 문자를 새로운 char형 배열에 저장하고 반복이 끝나면 널 문자를 붙인다. 이렇게 만든 배열과 내가 찾을 단어를 비교하여, 같다면 해당 단어의 첫

인덱스를 col에 저장하고 flag를 1로 저장한다. 같지 않다면 다음 단어로 넘어가는데, 내가 찾을 단어가 문장의 단어 중 일부분일 수 있기에 한 단어라도 한 문자씩 비교하도록 구현했다. 이렇게 구현한 search 함수에 인자로 전달한 flag와 col이 참조자 이므로 함수를 호출한 부분에서 사용하는 변수도 바뀐다. 함수의 인자로 전달하는 문장은 배열의 한 행이므로 flag가 1일 때의 행, 함수를 호출하면서 바뀐 열을 출력하면 해당 단어의 행과 열이 출력된다.

<change>

change 함수를 호출하기 전 search 함수를 호출하여 해당 단어가 문장에 있는지, 있다면 몇 번째 열인지를 저장하고 인자로 전달하기 때문에 change 함수는 flag가 0이라면 단어가 없으므로 함수를 종료하고, flag가 1이라면 단어가 있으므로 실행하도록 설정했다.
바꿀 단어와 바뀔 단어는 서로 길이가 다를 수 있으므로, 단어를 기준으로 바로 앞 문장까지 새로운 배열에 저장하여 널 문자를 붙여 한 문자열을 만들고, 단어의 바로 뒤부터 끝까지 새로운 배열에 저장하여 널 문자를 붙여 한 문자열을 만든다. 이렇게 앞 문장, 단어, 뒤 문장으로 구분한 후 또 새로운 배열에 앞 문장을 복사, 새로운 단어를 복사, 남은 뒤 문장을 복사하여 단어를 바꾼 새로운 문장을 저장한다. 앞 문장의 길이 + 뒤 문장의 길이 + 바뀔 단어의 길이는 새로 만든 문장의 길이이므로 해당 인덱스에 널 문자를 붙여 문자열로 저장했다. 원래의 문장이 저장된 행의 널 문자를 공백으로 대체 후 다시 원래의 배열에 새로 저장된 문장을 복사하고 널 문자를 붙여 새로운 문장을 다시 배열에 저장했다.

<insert>

insert 함수는 행, 단어, 해당 행의 문장을 인자로 전달받는다. insert도 change와 비슷한 방식으로 구현했는데, 인자로 전달받은 열을 기준으로 앞 문장을 복사하고 널 문자를 붙여 하나의 문자열로 만들고, 열의 바로 뒤부터 끝까지 문장을 복사하고 널 문자를 붙여 하나의 문자열을 만들어서 다시 새로운 배열에 앞 문장, 단어, 뒤 문장을 복사 후 널 문자를 붙여 단어가 삽입된 문장을 문자열로 만든다. 이렇게 만든 문자열을 원래 배열의 행에 복사하여 새로운 문장을 저장했다.

<delete>

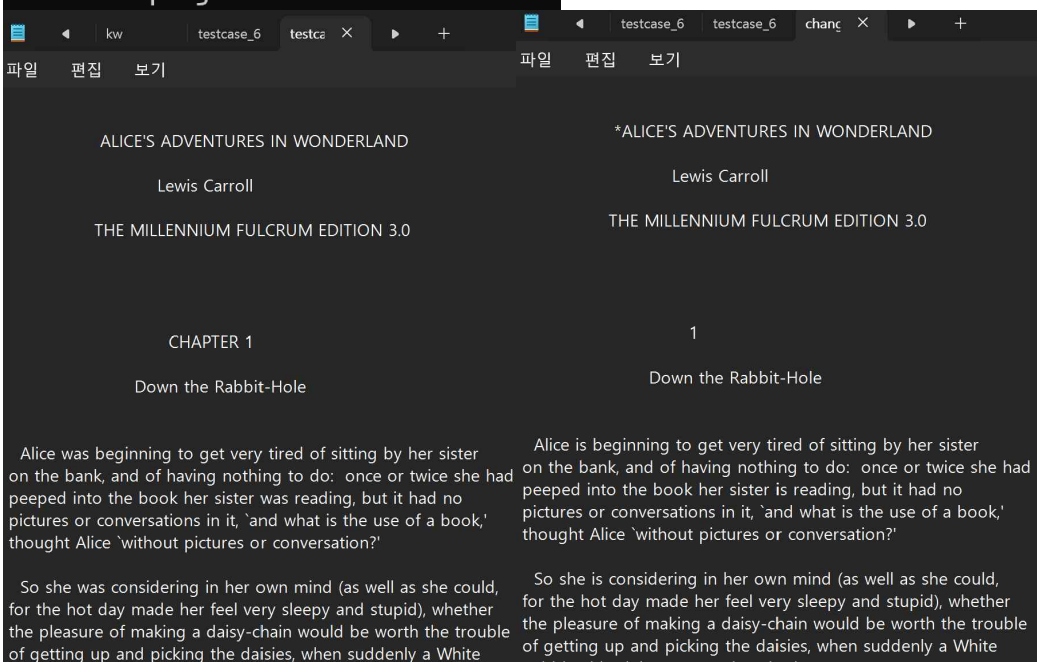
delete도 change와 마찬가지로 함수를 호출하기 전 search를 호출하므로 flag가 0이라면 실행하지 않고 1이라면 실행하도록 구현했다. change와 거의 비슷하게 해당 단어를 기준으로 앞 뒤 문장으로 나누어 널 문자를 붙여 새로운 2개의 문자열을 만든다. 그리고 search에 반환된 열을 기준으로 없앨 단어의 길이만큼의 문장을 공백으로 바꾼다(문장에서 없앨 단어 부분). 이렇게 공백 처리된 행에서 없앨 단어의 시작 인덱스부터 새로 저장한 뒷부분의 문장을 복사하여 단어를 없앤 문장을 저장한다.

<save>

새롭게 바뀐 문장을 파일에 저장하기 위해서 파일의 이름을 입력받은 후 ofstream의 open을 이용하여 새로운 파일을 연다. ofstream 객체 fout을 선언하여 2중 for 문과 fout <<을 이용하여 배열의 모든 행과 열을 출력하고 파일을 닫으면 새롭게 저장된 파일에 바뀐 문장이 저장된다.

<결과>

```
open ./testcase_6.txt
search glass
==='glass' search(3)===
(126, 6) (200, 26) (217, 32)
------------------------------
change was is
===change===
was -> is
------------------------------
insert 1 16 *
===insert===
* Inserted into (1, 16)
------------------------------
delete CHAPTER
===delete===
Delete CHAPTER
------------------------------
save ./changedtext.txt
===save===
Save the file as "./changedtext.txt"
------------------------------
exit
Exit the program
```



변경 전-> 변경 후

문장의 첫 행, 열을 0, 0이라고 하자. CHAPTER를 검색하면 CHAPTER의 위치인 10, 28이 출력되고 glass를 검색하면 문장의 모든 glass의 위치가 출력된다. change was is는 문장의 모든 was를 is로 바꾸는데, 새로 저장된 문서를 보면 모든 was가 is로 바뀜을 볼 수 있다. 또한 insert 1 16 *은 1, 16에 *을 삽입하는데 새로 저장된 문장의 1, 16에 별이 삽입된 것을 볼 수 있다.

<고찰>

파일 입출력에 익숙하지 않아서 ifstream, ofstream객체 모두 어떻게 사용할지 막막했다.

하지만 객체지향프로그램설계 수업에서 설명들은 내용으로 키보드와 모니터를 파일로 대체한다고 생각하고 문제에 접근하니까 익숙하게 다가왔다. ifstream에서 cin은 한 문자, getline은 한 문장씩 받아온다는 차이점만 잘 기억하면 나중에도 파일 입출력을 사용할 때 어렵지 않게 사용할 수 있겠다는 생각이 들었다.

<Assignment #2-7>

7. Write a calculator program that calculates a formula entered by the user. The calculator should support only integers and basic arithmetic operations (+, -, *, /). Write it so that it can handle parentheses. Utilize 'stack' to calculate the higher priority parts of the expression first. Implement and use the stack without using the stack library.
   - input numbers are between 0 and 255.
   - output number is between -2,147,843,648 and 2,147,483,647.
   - the number of operators is between 0 and 8.
   - the number of parentheses pair is between 0 and 8.

```
Enter the formula : 15 - ( 3 * 2 + 11 ) / 2
Result : 7
```

```
Enter the formula : 0
Result : 0
```

```
Enter the formula : 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9
Result : 45
```

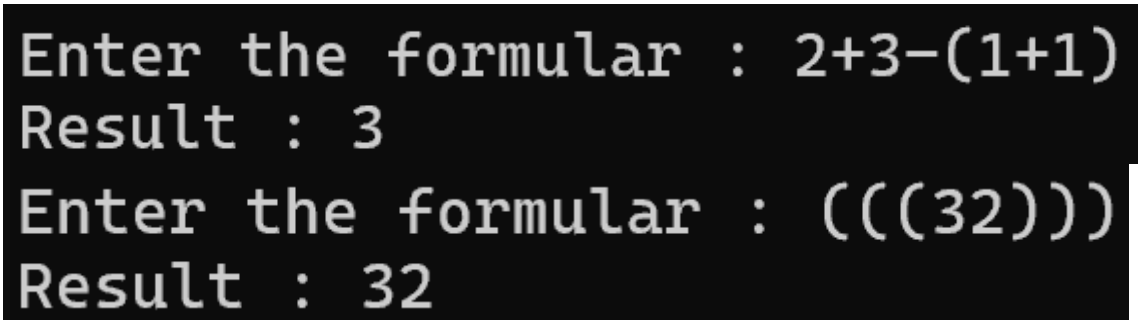
```
Enter the formula : (((((((( 255 ))))))))
Result : 255
```

```
Enter the formula : (((((((( 1 ) - 2) + 3) - 4) + 5) - 6) + 7) - 8) + 9
Result : 5
```

사칙연산을 입력하면 답을 출력하는 프로그램이다. 입력하는 수는 0~255의 정수이고 연산자는 최대 8개까지 사용할 수 있다. 스택을 사용하여 연산자의 우선순위를 결정해서 프로그램을 구성해야한다.

<알고리즘>
연산자와 피연산자를 저장하기 위해 스택 클래스를 정의했다. 스택 클래스의 멤버 변수는 스택을 가리키는 int ptr, 스택을 구성하는 int형 배열 arr을 가진다. 생성자에서는 ptr을 -1로 초기화했고 ptr을 통해서 스택의 값에 접근한다. 스택의 기본 함수 중 하나인 push는 스택에 값을 넣어주는 기능으로, ptr을 증가시키고 arr[ptr]에 값을 저장한다. 따라서 ptr은 제일 나중에 저장된 값의 인덱스를 가리킨다. 스택의 제일 위 값을 반환하고 삭제시키는 pop은 ptr이 0보다 작으면 스택이 비어있으므로 함수를 종료하고 0 이상이라면 arr[ptr]의 값을 저장하여 ptr을 하나 감소시키고 값을 반환한다. 스택의 값은 ptr로 접근하므로 ptr이 하나 줄어들면 값이 하나 없어진 것과 같은 기능을 할 수 있다. 스택의 제일 위 값을 삭제하지 않고 반환하는 peek 함수는 ptr은 항상 제일 나중에 저장된 값의 인덱스이므로 arr[ptr]을 반환하면 된다. 마지막으로 스택이 비어있는지 판단하는 empty 함수는 bool

형으로 선언하여 ptr이 0보다 작으면 스택이 비어있으므로 false, 아니라면 true를 반환하도록 했다.
사칙연산을 구성하는 함수는 총 4개로 되어있는데, 문자가 숫자인지 판단하는 digit, 연산자의 우선순위를 정하는 priority, 숫자를 계산하는 operate, 연산자와 피연산자를 스택에 넣고 우선순위에 따라서 계산을 구성하는 calculate 함수로 구성된다. 먼저, priority 함수는 +, -는 1을 반환, *과 /은 2를 반환하고 나머지는 0을 반환한다. operate 함수는 연산자에 따라서 +, -, *, /을 진행하는데, /연산을 진행할 때, 분모가 0이라면 exit(1)을 통해서 1을 반환하고 종료한다. 또한 다른 문자가 들어온다면 마찬가지로 exit(1)을 통해서 1을 반환하고 종료시킨다. digit은 인자로 전달되는 문자가 '0'보다 크고 '9'보다 작다면 true, 아니라면 false를 반환한다.
calculate 함수는 stack 객체 2개를 선언하여 하나는 피연산자를 저장하고 다른 하나는 연산자를 저장하는 용도로 사용한다. for 문을 통해서 I가 0부터 입력받는 사칙연산의 끝까지 반복하도록 한다. arr[i]가 빈칸이라면 continue를 통해서 다음 반복으로 넘어간다.
arr[i]가 여는 괄호'('라면 연산자의 스택에 저장한다.
digit 함수를 통해서 arr[i]가 숫자인지 판별하고 숫자라면 int temp를 0으로 초기화한다. while 문을 통해서 temp=(10*temp) + (arr[i]-'0')으로 저장한다. 사칙연산의 피연산자는 0~255의 숫자이므로 한 자리의 수가 아닐 수도 있고 arr[i]는 문자이므로 '0'을 빼주어 정수로 바꾸고 10*temp 더해주어서 두 자리나 세 자리의 정수도 temp에 저장되도록 했다. 이때 while마다 I를 증가시켜서 temp에 정수를 저장하므로 자릿수+1 만큼 I가 증가한다. 따라서 I를 하나 감소시켜서 자릿수만큼만 I가 증가하도록 했다. 이렇게 저장한 temp를 피연산자 스택에 넣어준다.
arr[i]가 닫는 괄호')'라면 while을 통해서 연산자 스택이 비어있지 않고 peek를 통해서 연산자 스택의 가장 위 연산자를 반환하고, 이 연산자가 여는 괄호가 아니라면 연산자와 피연산자 2개의 연산을 진행한다. 이때 연산자는 pop을 통해서 값을 반환하고 삭제한다. -와 /는 교환법칙이 성립하지 않으므로 피연산자의 가장 위에 저장된 수를 operate의 두 번째 인자로, 다음 수를 첫 번째 인자로 전달한다.
이렇게 반복을 진행할 때 괄호 안에 연산자가 괄호보다 우선순위가 높으므로 먼저 괄호의 연산을 진행한다. 반복을 진행하고 나면 연산자는 다 없어지고 여는 괄호만 남아있으므로 마지막에 pop을 통해서 여는 괄호를 없앤다.
arr[i]가 연산자라면 스택이 비어있지 않고 현재 스택의 저장된 가장 위 연산자의 우선순위가 현재 arr[i]보다 높거나 같다면 연산을 진행한다. 이때 연산자 스택의 연산자는 pop을 통해서 삭제한 후 연산을 진행한다. 이렇게 연산을 모두 마치면 피연산자의 스택에는 연산이 끝난 값만 남아있으므로 peek을 이용하여 결과를 반환한다.



<결과>
사칙연산을 문자로 입력하면 괄호 안의 연산을 먼저 처리, 곱셈·나눗셈, 덧셈·뺄셈의 순서대로 처리하여 값을 반환한다.
<고찰>
스택을 사용하라고 나와 있어서 어떻게 스택을 이용하여 사칙연산을 구현할지 고민했다. 먼저 계산할 연산자가 구분되어 있으므로 각 연산자에 순위를 매겨서 계산했는데, 괄호를 처리하는 부분에서 어려움을 느꼈다. 하지만 스택의 특성인 FILO와 값을 반환하고 삭제하는 pop을 잘 활용하여 닫는 괄호를 받는다면 여는 괄호 이후에 저장된 연산을 pop을 통해서 계산 후 연산자를 없애는 방식으로 사칙연산을 구성했다. 이 문제를 구현하고 공부하며 스택을 이런 방식으로도 이용할 수 있다는 것에 놀랐다.

8. Implement a 'Student' class with the members, as shown in the figure below, and create a program that performs actions based on the commands (insert, find, change, print, exit). The program should be able to store information of up to 10 students.
   - assignmentScore, examScore, attendance can range from 0 to 100
   - finalScore is calculated as 10% of attendance and 40% of assignmentScore and 50% of examScore
   - The name, studentID can be up to 100 characters
   - Case insensitive

Command	Format	Description
insert	insert [name] [studentID] [assignmentScore] [ExamScore] [attendance]	Add a new student object that stores the input information as its member variables through a constructor.
find	find [name]	If the name was found, print its information, otherwise print a 'not found' message.
change	change [name] [assignmentScore] [ExamScore] [attendance]	Change student [name]'s score information.
print	print	Print out the information of all stored students.
exit	exit	Exit the program.

```
class Student
{
private:
    char* name;
    char* studentID;
    double assignmentScore;
    double examScore;
    double attendance;
    double finalScore;

public:
    Student(char* name, char* id, double aScore, double eScore, double att);
    ~Student();
    bool isNameCorrect(char* name);
    void changeScores(double aScore, double eScore, double att);
    void print();
};
```

Input	Output
insert Kim 2024000001 80 80 100 insert Park 2024000002 50 75 80 insert Choi 2024000003 90 90 90 print find Lee find Kim change Kim 90 90 100 print exit	=====print===== Name : Kim Student ID : 2024000001 Final Score : 82 ---------- Name : Park Student ID : 2024000002 Final Score : 65.5 ---------- Name : Choi Student ID : 2024000003 Final Score : 90 ---------- =====find===== not found ---------- =====find===== Name : Kim Student ID : 2024000001 Final Score : 91 ---------- =====print===== Name : Kim Student ID : 2024000001 Final Score : 91 ---------- Name : Park Student ID : 2024000002 Final Score : 65.5 ---------- Name : Choi Student ID : 2024000003 Final Score : 90 ---------- Exit the program

<Assignment #2-8>

student 클래스를 사진의 포맷으로 구성한다. 이때 멤버 변수는 이름, id, 과제점수, 시험점수, 출석, 최종점수를 가진다. insert 커맨드는 이름, id, 과제점수, 시험점수, 출석을 입력하면 새로운 학생을 한 명 만들고 id와 점수들을 저장한다. find 커맨드는 이름을 입력하여 해당 학생의 정보를 출력하는데, 이때 이름에 맞는 학생이 없다면 not find를 출력한다. change 커맨드는 이름, 새로운 과제점수, 시험점수, 출석을 입력하면 해당 이름의

학생의 점수들을 새롭게 바꾼다. print는 저장된 모든 학생의 정보를 출력한다. exit는 함수를 종료한다.

<알고리즘>

student의 객체 중 이름과 id는 char형 포인터이므로 생성자에서 동적할당으로 메모리를 생성하고 직접 정의한 strcpy를 통해서 인자로 전달된 이름과 id로 초기화고 나머지 점수도 인자로 전달된 점수들로 초기화했다. 최종점수는 출석 10%, 과제 40%, 시험 50%의 비율로 결정되므로 이에 맞추어서 최종점수도 초기화했다.

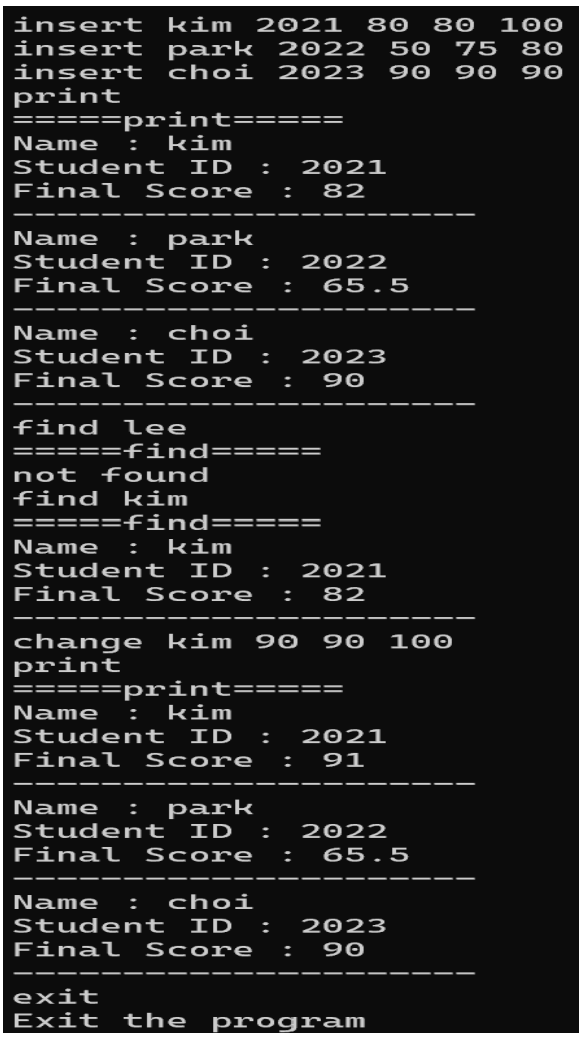
생성자에서 동적할당을 진행했으므로 소멸자에서 name과 ID에 할당된 메모리를 해제했다.

이름과 같은지 판단하는 isNameCorrect 함수는 입력받은 이름과 저장된 학생의 정보 중 맞는 이름이 있다면 false, 없다면 true를 반환하도록 구현했다. 찾는 이름과 저장된 이름의 대·소문자는 구분하지 않으므로 객체 멤버의 이름과 입력받는 이름을 모두 대문자라면 알파벳 대·소문자의 아스키 값 차이인 32를 더해서 소문자로 모두 바꾸었다. 이후 직접 정의한 strcmp를 이용하여 이름이 같다면 flase, 다르다면 true를 반환한다.

해당 학생의 점수를 바꾸는 changeScores함수는 입력받은 점수들을 모두 바꾸어 주면 된다.

최대 10명의 학생을 저장하므로 student형 객체 포인터 배열을 생성하여 각 배열의 요소에 학생들의 정보를 저장하도록 구현했다.

<결과>



kim, park, choi의 정보를 저장하고 출력하면 모든 학생의 정보가 나온다. find lee를 검색하면 lee의 이름을 가진 학생이 없으므로 not found가 출력되고 find kim을 출력하면 kim의 정보가 나온다. change kim 90 90 100을 입력하면 kim의 점수가 90 90 100으로 바뀌고 이에 따라서 최종점수도 바뀐다.

<Assignment #2-9>

9. Implement a program that stores and prints student information. To store student information, implement the Student Class below, and implement the School Class, which aims to manage multiple students' information. The Student Class should not be accessible outside of the School Class. Each class can only have the member variables shown in the figure below, and member functions can be implemented freely. The program should support four commands: new_student, sort_by_score, print_all, print_A_grade, print_B_grade, and exit.

   - The name, studentID can be up to 100 characters

```
class Studnet
{
private:
    char* name;
    char* studentID;
    double Score;
};
class School
{
private:
    class Student* student_list;
    int size = 0;
};
```

Command	Format	Description
new_student	new_student [name] [studentID] [Score]	Store the information of a new student in the school class.
sort_by_score	sort_by_score	Sort the student objects by Score in descending order.
print_all	print_all	Print information of all students stored in the School class.
print_A_grade	print_A_grade	Print information about students whose scores are in the top 30 percent
print_B_grade	print_B_grade	Print information for students whose scores are in the top 50% but are not A's
exit	exit	Terminate program

Input	Output
new_student Olivia 2013000005 95 new_student Emma 2013020103 55 new_student Amelia 2014100001 71 new_student Ava 2011020306 64 new_student Sophia 2013000004 80 print_all sort_by_score print_all exit	=====print===== Name : Olivia Student ID : 2013000005 Score : 95 ---------- Name : Emma Student ID : 2013020103 Score : 55 ---------- Name : Amelia Student ID : 2014100001 Score : 71 ---------- Name : Ava Student ID : 2011020306 Score : 64 ---------- Name : Sophia Student ID : 2013000004 Score : 80 ---------- =====print===== Name : Olivia Student ID : 2013000005 Score : 95 ---------- Name : Sophia Student ID : 2013000004 Score : 80 ---------- Name : Amelia Student ID : 2014100001 Score : 71 ---------- Name : Ava Student ID : 2011020306 Score : 64 ---------- Name : Emma Student ID : 2013020103 Score : 55

Input	Output
new_student Olivia 2013000005 95 new_student Emma 2013020103 55 new_student Amelia 2014100001 71 new_student Ava 2011020306 64 new_student Sophia 2013000004 80 print_A_grade print_B_grade exit	=====A grade===== Name : Olivia Student ID : 2013000005 Score : 95 ---------- =====B grade===== Name : Sophia Student ID : 2013000004 Score : 80 ---------- Exit the program

8번과 비슷한 student 클래스를 정의하고 이 student 객체를 관리하는 school 클래스를 정의한다. 이때 student 클래스는 school 클래스의 외부에서는 접근할 수 없다.
new_student는 이름, id, 점수를 입력하여 새로운 학생 정보를 저장하는 커맨드이다.
sort_by_score는 각 점수에 따라서 학생의 정보를 내림차순으로 정렬하는 커맨드이다.
print는 모든 학생의 정보를 출력, print_A_grade는 상위 30%를 출력, print_B_grade는 상위 30%~50%를 출력한다.
<알고리즘>
각 등급의 산출 퍼센트는 내림이다. 따라서 내림을 해주는 floor 함수를 정의하여 입력받은 double 형 변수를 int 형으로 변환 후 반환하여 내림을 구현했다.
먼저 student는 8번과 비슷하게 이름과 id는 동적할당 후 strcpy를 이용하여 초기화하고 점수도 입력받은 점수로 초기화했다. 하지만 student 클래스의 소멸자에서 동적 할당한

메모리를 해제하면 school 클래스에서 student의 멤버 변수에 접근하거나 관리할 때 제대로 동작하지 않을 수 있으므로 student에 이름과 id와 점수를 반환하는 함수를 만들어서 school 클래스의 소멸자에서 메모리를 해제했다.
school 클래스의 멤버 변수는 student 객체 포인터 student_list와 학생의 수인 size를 가진다. 따라서 생성자에서 객체 포인터에 동적할당을 사용하여 메모리를 할당했다.
소멸자에서는 아까도 언급한 것처럼 student의 생성자에서 할당한 메모리와 school 클래스의 생성자에서 할당한 메모리를 모두 해제했다. 학생의 정보를 추가하는 addStudent 함수는 student_list가 student 객체 포인터이므로 student_list[size]에 Student의 생성자를 통해서 객체를 저장하고 size를 하나 증가시켰다.
학생의 점수에 따라서 내림차순으로 정렬하는 sortStudent 함수는 버블정렬 알고리즘을 사용하여 객체 포인터 배열의 요소 중 j의 점수가 j+1의 점수보다 작다면 객체 자체를 교체하여 점수에 따라서 내림차순으로 학생이 정렬되도록 했다.
A등급을 출력하는 print_A_grade 함수는 아까 구현했던 내림함수에 size(저장된 학생의 수)*0.3을 전달하여 0부터 내림 된 값의 인덱스에 저장된 학생의 정보를 출력했고 print_B_grade 함수도 마찬가지로 size*0.3이 내림 된 값과 size*0.5가 내림 된 값의 인덱스에 저장된 학생의 정보를 출력했다.
<결과>

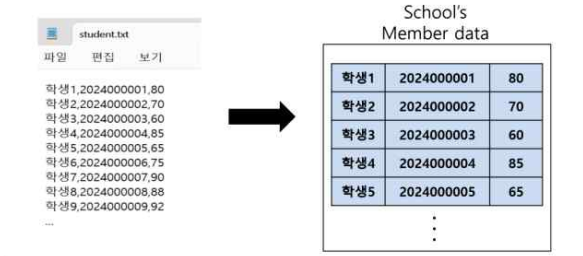
```
new_student kim 2021 95
new_student park 2022 55
new_student lee 2023 80
new_student choi 2024 71
print_all
=====print====
Name : kim
Student ID : 2021
Score : 95
-------------------------
Name : park
Student ID : 2022
Score : 55
-------------------------
Name : lee
Student ID : 2023
Score : 80
-------------------------
Name : choi
Student ID : 2024
Score : 71
-------------------------
sort_by_score
print_all
=====print====
Name : kim
Student ID : 2021
Score : 95
-------------------------
Name : lee
Student ID : 2023
Score : 80
-------------------------
Name : choi
Student ID : 2024
Score : 71
-------------------------
Name : park
Student ID : 2022
Score : 55
-------------------------
exit
Exit the program
```

new_student 커맨드를 이용하여 학생들의 정보를 저장한다. 이때 아직 정렬되지 않은 상태로 출력되는데, sort_by_score를 사용하여 내림차순으로 정렬한 후 출력하면 결과와 같이 내림차순으로 정렬되어 출력된다.
<고찰> student 클래스를 school 클래스를 통해서 접근해야 하는데 student 클래스의 생성자에서 할당한 메모리를 student 클래스의 소멸자에서 소멸했다. 하지만 이렇게 되면 임시 객체를 생성하여 school 객체를 생성할 때, student의 소멸자에서 메모리를 해제하므로 이름과 id가 깨져서 출력된다. 따라서 이름과 id를 반환하는 함수를 만들어 school 클래스의 소멸자에서 메모리를 해제하도록 구현하여 오류를 해결했다.

<Assignment #2-10>

10. Write a program to read a 'student.txt' file and store information about students in a school class.
    - text files separate columns with a delimiter (,) and rows with newlines
    - Every row has the same number of columns
    - The name, studentID can be up to 100 characters

    change the 'new_student' command to load the student's information via cin at problem 9 to the 'load_student' command to load the student's information via a file.

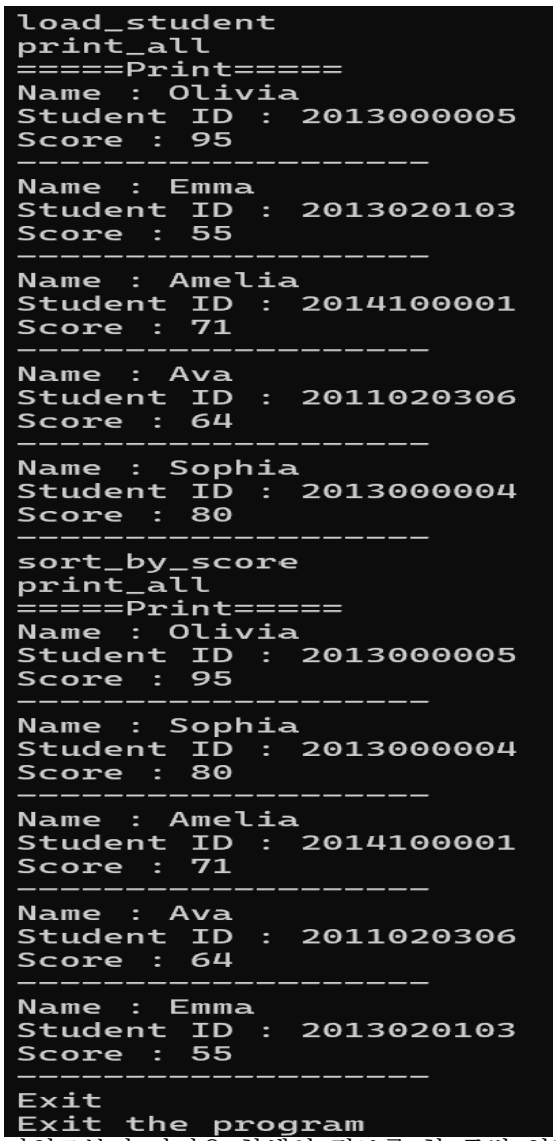


Input	Output
load_student print_all sort_by_score print_all Exit	=====print===== Name : Olivia Student ID : 2013000005 Score : 95 --- Name : Emma Student ID : 2013020103 Score : 55 --- Name : Amelia Student ID : 2014100001 Score : 71 --- Name : Ava Student ID : 2011020306 Score : 64 --- Name : Sophia Student ID : 2013000004 Score : 80 --- =====print===== Name : Olivia Student ID : 2013000005 Score : 95 --- Name : Sophia Student ID : 2013000004 Score : 80 --- Name : Amelia Student ID : 2014100001 Score : 71 --- Name : Ava Student ID : 2011020306 Score : 64 --- Name : Emma Student ID : 2013020103 Score : 55 --- Exit the program

school 클래스를 정의하여 파일로부터 학생의 이름, ID, 점수를 받아와서 모두 클래스의 객체에 넣고 점수대로 내림차순 정렬하는 프로그램이다.
<알고리즘>
클래스의 멤버 변수는 학생의 이름, id, 점수이고 이름과 id는 char *형, 점수는 int 형으로 선언했다. 마찬가지로 생성자에서 동적할당을 통해서 이름과 id에 메모리를 할당하고 점수는 이니셜라이저를 이용하여 0으로 초기화했다. 따라서 소멸자에서 할당한 메모리를 해제한다. 나중에 점수를 출력해야 하므로 score를 반환하는 함수도 만들었다. 객체에 학생의 정보를

저장하는 load 함수는 인자로 이름, id, 점수를 받는데 delete를 통해서 이전에 저장된 이름과 id의 메모리를 해제하고 다시 메모리를 할당한 후 인자로 전달된 이름과 id, 점수로 초기화한다.
파일에서 입력을 받아와야 하므로 ifstream 객체인 fin과 school형 포인터 배열 stu를 선언했다.
먼저 load_student를 입력하면 함수로부터 파일을 받기 위해서 open을 통해 입력받을 파일을 열고 getline을 통해서 char *형 temp에 한 문장씩 저장했다. 파일에서 받아온 문장은 ,를 기준으로 이름, id 점수가 나뉘어져 있으므로 한 문장에서 공백을 기준으로 이름, id, 점수를 구분할 separate 함수를 구현했다.
separate의 인자는 내가 받아올 문장, char*형 참조자로 선언한 이름과 id, int 형 참조자인 점수로 구성된다. ,를 기준으로 나눈 단어, id, 점수를 저장하기 위해서 행의 크기가 3인 char 형 2차원 배열을 선언했다. for문을 통해서 I가 0부터 문장의 길이-1까지 반복하는데 행을 구성하는 int 형 변수 j와 열을 구성하는 int 형 변수 ind를 선언하여 arr의 한 행의 처음부터 저장되도록 했다. arr[j][ind]에 문장의 값을 복사하면서 ind를 증가시킨다. 만약 temp[i]가 ,라면 arr[j][ind]에 널문자를 저장하여 한 행을 완성하고 j를 하나 증가시켜 다음 행으로, ind를 0으로 초기화하여 다음 행의 첫 인덱스부터 저장되도록 만들었다.
공백이라면 continue를 사용하여 다음 문자로 넘어가도록 했다.
이렇게 한 문장에서 separate 함수로 나눈 단어, id, 점수를 stu 배열의 요소인 school 객체를 생성했다.
점수에 따라서 내림차순으로 정렬하는 sort_by_score 함수는 9번 문제와 동일하게 버블소트를 사용하여 정렬하는데, school 클래스에서 정의했던 점수를 반환하는 함수를 이용하여 비교하고 정렬했다.
exit 커맨드가 입력되면 프로그램을 종료해야 하므로 동적으로 할당한 메모리를 해제해야 한다. 객체 포인터 배열에 저장된 학생의 수만큼 반복하며 메모리를 해제하는데, stu[i]는 하나의 객체이므로 delete[]가 아닌 delete를 사용해서 해제해야 한다.

<결과>



파일로부터 가져온 학생의 정보를 한 줄씩 입력받아 이름, id. 점수로 구분하여 객체의 멤버로 저장했다. sort_by_score를 통해서 내림차순으로 정렬하고 출력하면 결과화면처럼 점수에 따라서 내림차순으로 학생들이 정렬되어 출력된다.

<고찰>

exit를 입력하면 프로그램이 종료하므로 프로그램이 종료하기 전에 동적으로 할당한 메모리를 모두 해제하고 종료해야 한다. 따라서 delete[]를 사용하여 객체의 메모리를 해제하고 종료하려고 했으나 메모리 누수로 인해 프로그램이 비정상적으로 종료되는 문제가 발생했다. delete는 객체의 메모리를 해제하고 delete[]는 배열의 할당한 메모리를 해제한다. c++에서는 배열에 메모리를 동적으로 할당할 때 배열 자체의 메모리 4 Byte를 추가하여 메모리를 할당하고 delete[]는 이를 확인하고 메모리를 해제한다. 하지만 내 코드의 객체 포인터 배열의 한 요소는 객체이므로 delete를 사용해야 제대로 메모리가 해제된다. 따라서 delete를 사용함으로써 문제를 해결했다.

<Assignment #2-11>

11. Write a program that uses a linked list to store integer values. The program has three commands that are insert, delete, and exit. You can see the details of each command in the table attached below. Please note that for the add command, the linked list must insert the input value and sort the list in descending order simultaneously. And each time an insert command or a delete command is entered, print all the values in the list in order.

Command	Format	Description
insert	insert [number]	Add value to a Linked list and prints all the values in the list.
delete	delete [number]	Delete values from a Linked list and print all values in the list. (Delete all duplicate values.)
exit	exit	Exit the program.

Input	Output
insert 1	Linked list : 1
insert 2	Linked list : 2 -> 1
insert 3	Linked list : 3 -> 2 -> 1
insert 4	Linked list : 4 -> 3 -> 2 -> 1
insert 5	Linked list : 5 -> 4 -> 3 -> 2 -> 1
insert 6	Linked list : 6 -> 5 -> 4 -> 3 -> 2 -> 1
delete 4	Linked list : 6 -> 5 -> 3 -> 2 -> 1
delete 1	Linked list : 6 -> 5 -> 3 -> 2
exit	Exit the program

연결리스트를 구현하여 insert를 입력하면 연결리스트에 넣을 숫자를 입력받아 집어넣는다. 이때 숫자는 내림차순으로 집어넣어야 한다. delete를 입력하면 숫자를 입력받고 해당 숫자를 지워 나머지 리스트를 연결하는 프로그램이다.

<알고리즘>

클래스를 이용하여 연결리스트를 구현했다. 먼저 각 연결리스트를 구성하는 데이터, 다음 연결리스트를 가리키는 포인터를 구조체(node)로 정의했다. 각 연결리스트를 구성하는 요소를 노드라고 칭하겠다. 연결리스트의 멤버 변수는 맨 앞을 가리키는 head, 맨 뒤를 가리키는 tail을 선언했다. 생성자에서는 객체 생성 시 아무것도 없으므로 head와 tail을 nullptr로 초기화했다. 노드를 추가하는 add 함수는 임시 노드인 node* temp와 추가할 노드인 cur을 선언했다. 만약 head가 nullptr이라면 리스트가 비어있다는 뜻이므로 temp의 data를 입력받은 수로 초기화하고 빈 리스트에 노드를 추가하면 하나밖에 없으므로 head와 tail을 모두 temp로 초기화 후, tail의 next는 pullptr로 초기화한다.

리스트에 하나 이상의 값이 있다면 cur의 데이터를 입력받은 수로 초기화하고 temp를 head로 저장한다. 내림차순으로 정렬하여 리스트의 값을 삽입해야 하므로 두 가지의 케이스로 나누어 구현했다. 먼저, cur의 데이터를 n으로 저장, temp는 head로 저장한다. 만약 temp의 데이터가 n보다 작다면 내림차순으로 구현해야 하므로 cur의 다음 노드를 head로 지정, head를 cur로 지정한다. temp의 데이터가 n보다 크다면 temp의 다음 노드가 nullptr이 아니고 temp의 다음 노드의 값이 n보다 크거나 같다면 temp=temp->next를

사용하여 temp를 다음 노드로 옮긴다. 따라서 temp는 내림차순으로 삽입해야할 노드의 바로 앞 노드가 되고, cur->next=temp->next로 초기화하여 내가 저장할 노드와 temp의 다음 노드를 연결하고 temp->next=cur로 초기화하여 temp와 내가 저장할 노드를 연결한다.
따라서 temp와 다음 노드 사이에 내가 저장할 노드가 삽입된다.
입력받을 값의 노드를 삭제하는 함수인 deleteNode는 내가 삭제할 노드의 바로 이전 노드를 찾아서 삭제하는데 head부터 반복한다.
여기서 flag는 내가 삭제할 노드의 이전 노드를 저장한다.
첫 번째로 flag의 다음 노드가 nullptr이 아니고 flag의 데이터가 n이라면 (2개 이상의 노드가 존재하고 내가 지울 노드가 맨 앞의 노드라면) head=head->next로 저장하여 head를 다음 노드로 옮기고 flag는 이전의 head이므로 flag의 메모리를 해제한다.
두 번째는 flag의 다음 노드가 nullptr이 아니고 flag의 다음 노드의 데이터가 n이라면 (2개 이상 존재) 삭제할 노드의 다음 노드를 임시 노드인 temp에 저장, flag->next를 temp->에 저장한다(이전 노드의 다음 노드를 삭제할 노드의 다음 노드로 저장). 그리고 temp의 메모리를 해제한다.
마지막으로는 flag->next가 nullptr이고 flag의 데이터가 n이라면 (원소가 1개) head와 tail을 nullptr로 초기화하여 리스트를 비운다.
if와 else if를 사용하여 이 세 가지 상황이 아니라면 flag를 다음 노드로 옮겨서 반복하도록 구현했다.
<결과>

```
insert 6
Linked list : 6
insert 9
Linked list : 9->6
insert 3
Linked list : 9->6->3
insert 4
Linked list : 9->6->4->3
insert 7
Linked list : 9->7->6->4->3
delete 9
Linked list : 7->6->4->3
delete 4
Linked list : 7->6->3
delete 3
Linked list : 7->6
exit
Exit the program
```

값을 넣을 때 내림차순이 되도록 맞는 위치에 삽입, delete로 값을 삭제하고 나머지 리스트를 다시 연결하여 출력하도록 했다.
<고찰>
처음에 문제를 제대로 읽지 않아서 리스트를 연결하고 내림차순으로 정렬하여 구현 과정에서

힘이 들었다. 문제를 다시 읽어보고 내림차순으로 연결리스트에 삽입하도록 바꾸었다. 따라서 삽입하는 과정에서 리스트가 비어있을 때, 리스트의 노드가 1개 이상이고 넣을 값이 head보다 클 때, 리스트의 노드가 1개 이상이고 넣을 값이 haed보다 작을 때로 나누어서 삽입했다. 앞으로는 코드를 구현하기 전에 제대로 문제를 이해하고 의도를 파악하여 구현해야겠다.

<Assignment #2-12>

12. Write a program that stores English words (maximum size of a word is 15) in alphabetical order(case insensitive) like the English dictionary. To store the words, the program uses the 2D linked list as a data structure the first order of the list represents an alphabet and the second order or list contains the words starting with the alphabet. The input of the program is given as input.dat and the output must be displayed to the standard output. The examples of input and output are given as follows:

Input format (Input.dat)

```
Apple
Mountain
Candle
Elephant
Notebook
Ocean
Tiger
Butterfly
Guitar
Laptop
River
Sunshine
Diamond
...
```

Output

```
A : Adult -> Advantage -> After -> Apple
B : Background -> Beak -> Beard -> Butterfly
C : Candle -> Cardboard
D : Desert -> Diamond -> Door
...
```

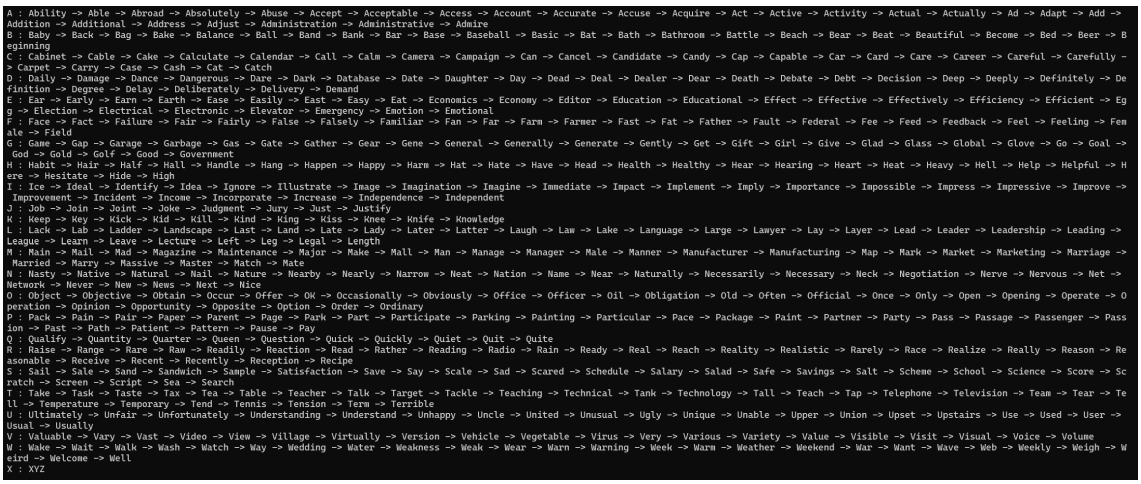
2차원 연결리스트를 구현하여 파일로부터 받아온 문장을 각 알파벳에 해당하는 단어끼리 묶어서 사전 순으로 출력하는 프로그램이다.

<알고리즘>

11번과는 다르게 2차원 연결리스트이므로 다음 노드를 가리키는 포인터인 next와 아래 노드를 가리키는 포인터인 under와 단어는 최대 15자리이므로 16 크기의 char형 배열을 선언했다. 알파벳의 개수는 26개이므로 멤버변수로 node* head[26]을 선언하여 각 알파벳에 해당하는 문자를 연결할 수 있도록 선언했다. 연결리스트에 값을 넣어주는 add 함수는 A는 0, B는 1, C는 3..의 순서로 배열에 할당할 것이므로 str[0]-'A'를 통해서 idx변수에 해당 문장이 들어갈 배열의 순서를 저장하고 11번에서 사용한 것과 마찬가지로 해당 행의 연결리스트에 값을 넣어주었다.

파일에서 읽어온 알파벳들은 사전 순으로 정렬되어 있지 않으므로 사전순으로 정렬하는 함수를 정의했다. I가 0부터 26까지 반복하며 head가 nullptr이면 해당 알파벳의 함수가 없으므로 수행하지 않고 비어있지 않은 연결리스트만 수행하도록 했다. I행 연결리스트의 head부터 반복하며 한 노드의 문자열과 뒤 노드의 문자열을 strcmp로 비교하여 값이 양수라면 앞 노드의 문자열이 사전 순으로 뒤에 있음을 의미하므로 strcpy를 통해 값을 교체하여 사전 순으로 정렬했다. 이 정렬 또한 버블소트의 방식을 사용했다.

<결과>



파일에서 받아온 단어들을 각 알파벳끼리 묶고 사전 순으로 정렬하여 출력했다. 이때, 해당 알파벳의 단어가 없다면 빼고 출력한다.

<고찰>

해당 알파벳의 단어가 없다면 빼고 출력하기 위해서 어떤 방식으로 앞의 알파벳을 출력해야할까 고민했다. 그러던 중 같은 알파벳으로 시작하는 단어만 묶어서 출력하는 프로그램이므로 리스트의 가장 첫 번째 값의 첫 인덱스는 해당 알파벳이므로 이를 이용하여 출력하면 단어가 있는 리스트만 출력할 수 있다는 생각이 들어서 이런 방식으로 구현했다.