Object Oriented Programming assignment2

설계 분반: 2 (신동화 교수님)
실습 분반: 1 (신동화 교수님)
학번: 2023202032
이름: 남호준

1.

문제 설명

1번 문제는 5~20 크기를 입력받으면 배열에 난수를 생성하고, 최댓값과 최솟값을 출력, 그리고 그 값의 위치 주소를 출력하는 프로그램이다.

알고리즘

Algorithm:

Initialize *init* with the current time as the seed value
Set the seed for random number generation using *init*
Declare variables *input*, *min*, and *max* as integers, initialized to 101 and -1 respectively
Declare variables *min_index* and *max_index* as integers, initialized to 0
Output "Size of the array : " to request input
Take user input and store it in *input*
Dynamically allocate an integer array of size *input* and assign it to *arr*
Output "Random numbers :"
for each integer *i* from 0 to *input - 1* **do**
    Generate a random number between 0 and 100 and assign it to *arr[i]*
    Output the generated random number
end for
for each integer *i* from 0 to *input - 1* **do**
    if *min* is greater than *arr[i]* **then**
        Update *min* to *arr[i]*
        Update *min_index* to *i*
    end if
    if *max* is less than *arr[i]* **then**
        Update *max* to *arr[i]*
        Update *max_index* to *i*
    end if
end for
Output "Minimum value : *min*, Address: &arr[*min_index*]"
Output "Maximum value : *max*, Address: &arr[*max_index*]"
Deallocate memory allocated for *arr*
return 0

결과화면

```
Size of the array : 10
Random numbers : 49 59 59 67 97 19 15 38 28 3
Maximum value : 3        , Address: 00000201B67E6FE4
Maximum value: 97        , Address: 00000201B67E6FD0
```

가장 작은 값인 3, 최댓값 97이 출력되고 인덱스 차이가 4*5=20 차이 나는 16진수의 14만큼 차이가 났다.

고찰

1차 과제에서 배웠던 static_cast를 활용하여 시드값을 초기화 하는데 이용하였다. 최솟값, 최댓값을 난수 범위 밖의 값을 입력하여 배열을 선형 순회하면서 최댓값과 최솟값을 업데이트하였다. 또한, 그 인덱스를 저장하여 배열에 더해 출력하여 그 주소를 출력하였다.

2.

문제 설명

2번 문제는 10개의 수를 난수로 채운 뒤 출력하고 1, 2을 입력하여 퀵 소트 또는 머지 솔트를 실행시키고, 탐색값을 입력하여 찾거나 없을 경우 삽입하여 배열을 다시 출력하는 프로그램이다.
이때, 탐색 알고리즘은 이분 탐색을 이용하여 탐색한다.

알고리즘

Algorithm 1 Main Function

```
Function Declarations:
procedure QUICK_SORT(arr, st, cn)        ▷ Quick sort function declaration
    procedure MERGE_SORT(arr, st, cn)    ▷ Merge sort function declaration
        procedure MERGE(arr, st, cn)     ▷ Merge function declaration
            procedure MY_INSERTION(arr, cn, input)   ▷ Insertion sort function declaration
                Declare array tmp of size 10   ▷ Temporary array declaration
                init ← current time            ▷ Initialize seed value
                srand(init)                    ▷ Set seed value
                Declare variables: arr (dynamic array of size 11), input, st = 0, cn = 9   ▷ Initialize array and variables
                Output "Random values : "      ▷ Generate random values
                for i from 0 to 9 do
                    arr[i] ← random number between 0 and 100   ▷ Assign random values
                    Output arr[i]              ▷ Output generated values
                end for
                Output "Select sorting method (1: Quick Sort, 2: Merge Sort) : "   ▷ Select sorting method
                Take user input and store in input
                if input == 1 then
                    Call QUICK_SORT(arr, 0, 10)   ▷ Perform quick sort
                    Output "Sorted numbers (Quick Sort): "
                else if input == 2 then
                    Call MERGE_SORT(arr, 0, 10)   ▷ Perform merge sort
                    Output "Sorted numbers (Merge Sort): "
                end if
                for i from 0 to 9 do
                    Output arr[i]              ▷ Output sorted array
                end for
                Output "Enter a value to search : "   ▷ Perform binary search
                Take user input and store in input
                mid ← (st + cn)/2
                while True do
                    if st > cn then
                        Call MY_INSERTION(arr, cn, input)   ▷ Call insertion sort
                        Output "Updated numbers:"
                        for i from 0 to 10 do
                            Output arr[i]      ▷ Output updated array
                        end for
                        Break
                    else if input == arr[mid] then
                        Output "Searched number index : mid"
                        Break
                    else if input > arr[mid] then
                        st ← mid + 1
                        mid ← (st + cn)/2
                    else if input < arr[mid] then
                        cn ← mid − 1
                        mid ← (st + cn)/2
                    end if
                end while
```

Algorithm 2 Merge Sort

```
procedure MERGE_SORT(arr, st, cn)                    ▷ Merge sort function definition
    if cn == (st + 1) then
        Return                                                          ▷ Base condition
    end if
    mid ← (st + cn)/2                                                ▷ Calculate middle value
    Call MERGE_SORT(arr, st, mid)                                        ▷ Recursive call
    Call MERGE_SORT(arr, mid, cn)                                        ▷ Recursive call
    Call MERGE(arr, st, cn)                                          ▷ Call merge function
end procedure
procedure MERGE(arr, st, cn)                                 ▷ Merge function definition
    mid ← (st + cn)/2                                                ▷ Calculate middle value
    idx1 ← st                                                        ▷ Initialize start point
    idx2 ← mid                                                       ▷ Initialize end point
    for i from st to cn − 1 do
        if idx2 == cn then
            tmp[i] ← arr[idx1 + +]                          ▷ Case when the second array ends
        else if idx1 == mid then
            tmp[i] ← arr[idx2 + +]                           ▷ Case when the first array ends
        else if arr[idx1] ≤ arr[idx2] then
            tmp[i] ← arr[idx1 + +]                                   ▷ Compare values and insert
        else
            tmp[i] ← arr[idx2 + +]                                   ▷ Compare values and insert
        end if
    end for
    for i from st to cn − 1 do
        arr[i] ← tmp[i]                ▷ Move from temporary array to original array
    end for
end procedure=0
```

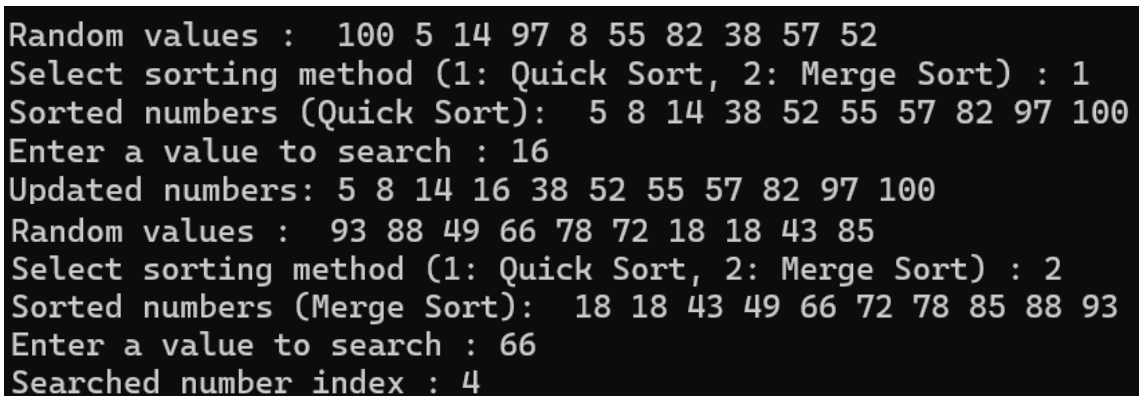
Algorithm 3 Quick Sort

```
procedure QUICK_SORT(arr, st, cn)                    ▷ Quick sort function definition
    if cn ≤ st + 1 then
        Return                                                          ▷ Base condition
    end if
    pivot ← arr[st]                                                  ▷ Set pivot value
    id1 ← st + 1                                                     ▷ Initialize index
    id2 ← cn − 1                                                     ▷ Initialize index
    while True do
        while id1 ≤ id2 and arr[id1] ≤ pivot do
            id1 + +                                                  ▷ Move index
        end while
        while id1 ≤ id2 and arr[id2] > pivot do
            id2 − −                                                  ▷ Move index
        end while
        if id1 > id2 then
            Break                                                   ▷ Break condition
        end if
        Swap arr[id1] and arr[id2]                          ▷ Compare and swap values
    end while
    Swap pivot and values smaller than pivot                ▷ Swap pivot and values
smaller than pivot
    Call QUICK_SORT(arr, st, id2)                                        ▷ Recursive call
    Call QUICK_SORT(arr, id2 + 1, cn)                                    ▷ Recursive call
end procedure
```

Algorithm 4 Insertion Sort

```
procedure MY_INSERTION(arr, cn, input)               ▷ Insertion sort function
definition
    for i from 10 to cn + 1 do
        arr[i] ← arr[i − 1]                                              ▷ Move elements
    end for
    arr[cn + 1] ← input                                                  ▷ Insert value
end procedure
```

결과화면



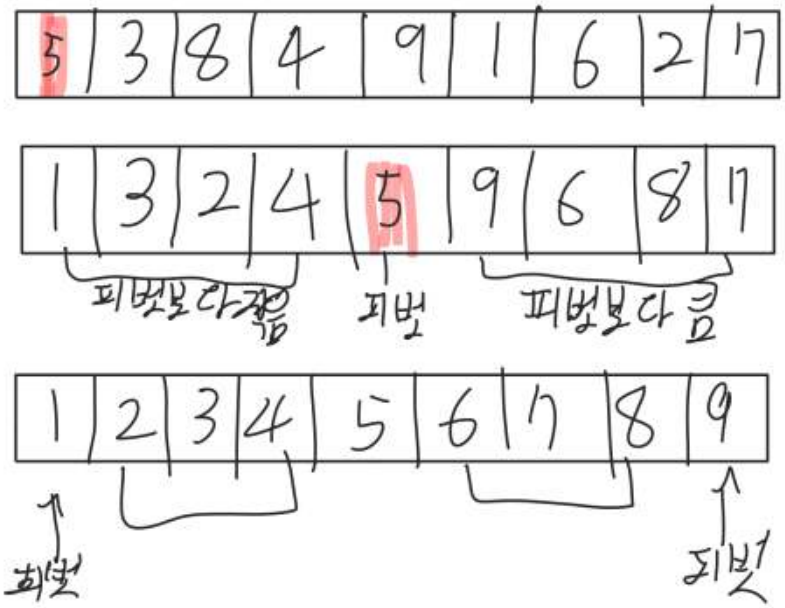
첫 번째 결과는 퀵 소트를 실행, 16을 탐색하여 없는 16을 정렬된 배열의 사이에 삽입되어 14와 38 사이에 들어가 정렬된 배열이 출력되었다.

두 번째 결과는 머지 솔트를 실행, 66이라는 탐색값을 입력하고 탐색하여 그 인덱스인 4를 출력하였다.

고찰

퀵 소트와 머지 솔트, 이분 탐색 알고리즘은 탐색과 정렬에서 높은 최적화 알고리즘 성능을 내는 것으로 알고 있고, 실제로 많이 사용되는 알고리즘이라고 알고 있다. 이를 실제로 찾아보고, 탐구하고 구현하면서 이해를 높이게 되었다.
공부하면서 알아본 두 정렬과 탐색 알고리즘의 시간 복잡도는 아래와 같다.
구현 중, 이분 탐색 알고리즘의 종료 조건에 대해 여러 케이스를 알아보면서 왜 start와 end가 크로스 될 때 종료되어야 하는지 알아보면서 재귀 또는 loop문의 종료 조건을 알아보는 중요한 시간이 되었다.

퀵 정렬



순환 호출의 횟수

k: $n = 2^k \Rightarrow k = \log_2 n$

비교 연산 과정: m $\therefore O(n) = m \cdot k = n \log n$

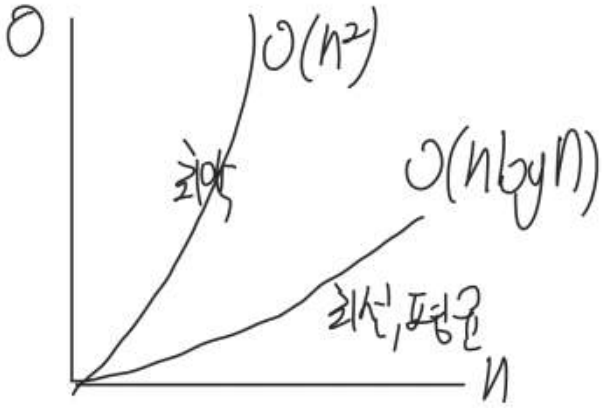
최악의 경우 (불균일 분할)

순환호출 횟수

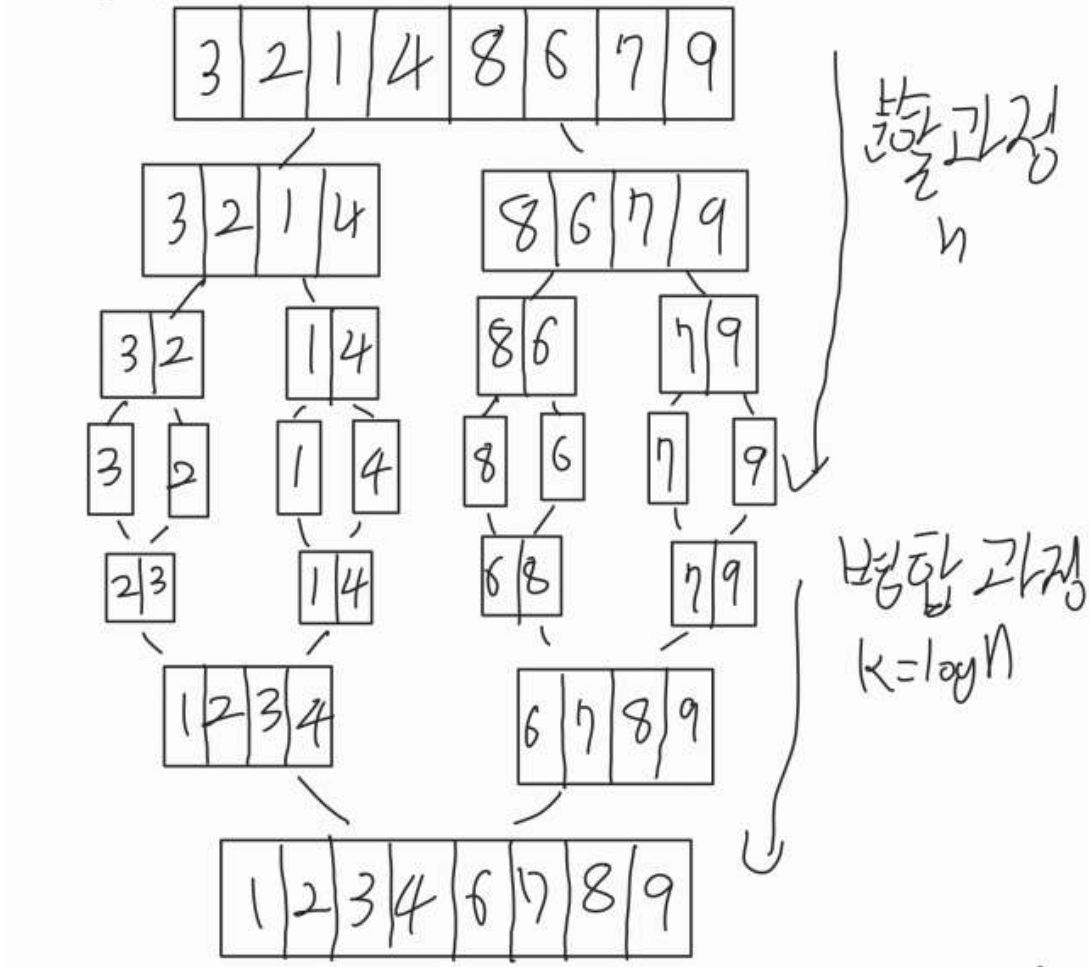
n

비교 연산과정: n

$\therefore O(n) = n \cdot n = n^2$

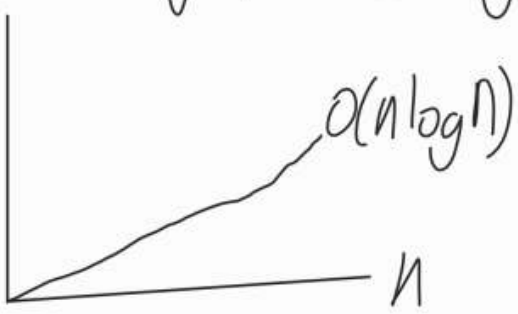


병합 정렬



비교횟수 : $k \times n = n \log n$ 이동횟수 : $2n \log n$

$\therefore$ 이동 + 비교 $= 3n \log n \approx O(n \log n)$



머지 솔트의
평균 = 최악 = 최선
은 같다.

3.

문제 설명

3번 문제는 10x10 크기의 행렬을 생성하고 각 행마다 내림차순으로 정렬 후 합계와 출력, 각 행의 합끼리 비교하여 오름차순으로 정렬 후 출력, 이때 각 줄을 합과 함께 이동시켜 출력하는 프로그램이다.
이때, 행렬은 이중 포인터에 동적할당을 이용하여 2차원 배열을 생성해서 만들어야 한다.

알고리즘

Algorithm 2 Main Function

```
1: Initialize init with the current time as the seed value
2: Create and initialize a 10 × 10 2D array mat
3: Allocate memory for an array sum to store the sum of each row
4: Calculate the sum of each row in mat
5: Print the original matrix
6: Sort rows in descending order based on their elements
7: Print the sorted matrix and row sums
8: Sort rows by sum in ascending order
9: Print the sorted matrix and row sums
10: Deallocate memory
```

Algorithm 3 Functions

```
1: procedure PrintMatrix(mat)
2:     for i in 1 to 10 do
3:         for j in 1 to 10 do
4:             Print mat[i][j]
5:         end for
6:         Print a newline
7:     end for
8: end procedure
9: procedure PrintMatrixWithSum(mat, sum)
10:    for i in 1 to 10 do
11:        for j in 1 to 10 do
12:            Print mat[i][j]
13:        end for
14:        Print | Sum: sum[i]
15:        Print a newline
16:    end for
17: end procedure
18: procedure SortRowsDescending(mat)
19:    for i in 1 to 10 do
20:        Sort mat[i] in descending order
21:    end for
22: end procedure
23: procedure SortRowsBySumAscending(mat, sum)
24:    for i in 1 to 9 do
25:        k ← i
26:        for j in (i + 1) to 10 do
27:            if sum[k] > sum[j] then
28:                Swap mat[k] and mat[j]
29:                Swap sum[k] and sum[j]
30:            end if
31:        end for
32:    end for
33: end procedure
```

결과화면

```
Original Matrix:
30      81      90      25      66      8       60      47      37      18
89      9       0       97      28      88      36      96      17      54
100     93      93      8       5       33      21      60      73      1
65      8       89      12      63      70      8       64      61      93
89      51      58      82      74      35      15      61      56      72
33      19      86      40      34      36      0       98      11      21
74      44      42      57      89      30      15      35      91      55
6       66      92      93      17      23      37      5       14      5
82      81      64      92      28      80      58      96      89      33
62      36      44      52      51      74      67      68      43      26

Sort of Row (Descending Order):
90      81      66      60      47      37      30      25      18      8       | Sum: 462
97      96      89      88      54      36      28      17      9       0       | Sum: 514
100     93      93      73      60      33      21      8       5       1       | Sum: 487
93      89      70      65      64      63      61      12      8       8       | Sum: 533
89      82      74      72      61      58      56      51      35      15      | Sum: 593
98      86      40      36      34      33      21      19      11      0       | Sum: 378
91      89      74      57      55      44      42      35      30      15      | Sum: 532
93      92      66      37      23      17      14      6       5       5       | Sum: 358
96      92      89      82      81      80      64      58      33      28      | Sum: 703
74      68      67      62      52      51      44      43      36      26      | Sum: 523

Sort of Sum (Ascending Order):
93      92      66      37      23      17      14      6       5       5       | Sum: 358
98      86      40      36      34      33      21      19      11      0       | Sum: 378
90      81      66      60      47      37      30      25      18      8       | Sum: 462
100     93      93      73      60      33      21      8       5       1       | Sum: 487
97      96      89      88      54      36      28      17      9       0       | Sum: 514
74      68      67      62      52      51      44      43      36      26      | Sum: 523
91      89      74      57      55      44      42      35      30      15      | Sum: 532
93      89      70      65      64      63      61      12      8       8       | Sum: 533
89      82      74      72      61      58      56      51      35      15      | Sum: 593
96      92      89      82      81      80      64      58      33      28      | Sum: 703
```

첫 출력엔 생성한 행렬 출력

다음 출력엔 행 별 내림차순과 합 출력

마지막 출력엔 각 행별 합으로 오름차순한 결과 출력

고찰

행끼리 위치를 바꿀 때 어떻게 해야할 지 고민하던 중, 각 행별로 정렬이 되어 있다는 것을 깨닫고 각 행 별로 같은 열 인덱스끼리 교환하는 알고리즘을 더하여 교환이 일어날 시 각 행, 열끼리 스왑하도록 만들었다.

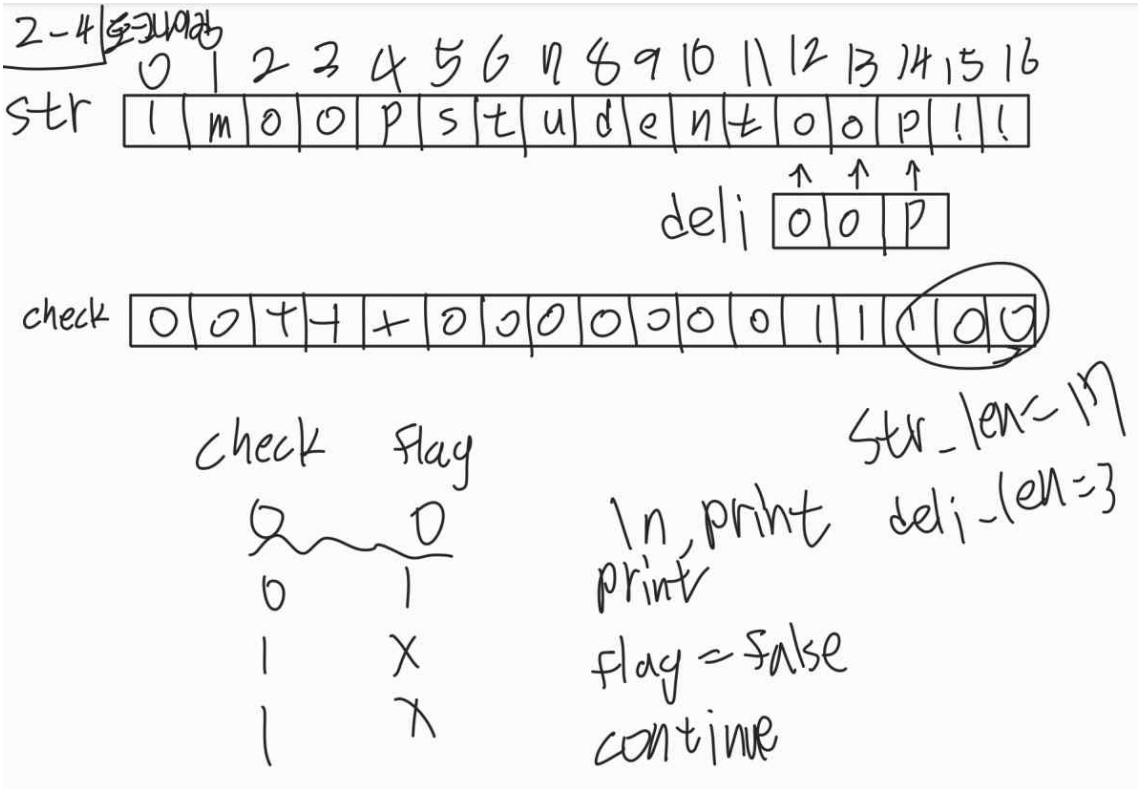
이것이 문제의 의도라고 생각하여 이러한 방식을 사용했다.

4.

문제 설명

4번 문제는 문장을 입력받고, 구분자를 입력받아 구분자로 문장을 자르고, 구분자로 나눠진 부분을 개행으로 나눠 출력하는 프로그램이다.
구분자는 10글자까지 입력받을 수 있고, 문자열은 100글자까지 된다.

알고리즘



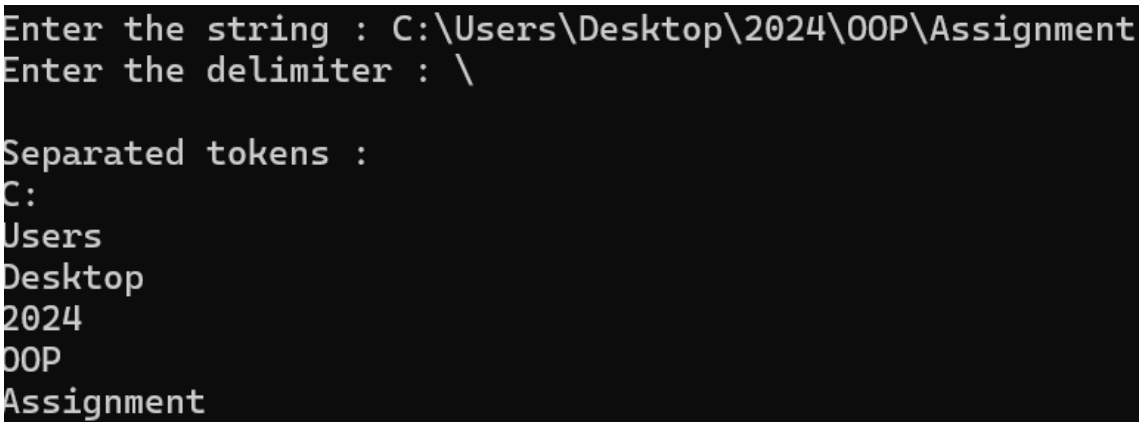
위와 같이 핵심 알고리즘은 구분자 배열을 입력받은 문자열 배열에 한 칸씩 비교해가며 같다고 판단될 시 check 배열에 체크 표시를 하여 출력 시 제외하도록 만들었다. check 배열과 flag를 사용하여 경우의 수에 따라 명령을 인코딩하여 구분하였다.

아래는 전체 코드의 수도 코드이다.

Algorithm 1 Separate String

```
1: Declare array input_string of characters of size 101 ▷ To store input string
2: Declare array delimiter of characters of size 11 ▷ To store delimiter
3: Declare array check of booleans of size 100 ▷ To mark positions of delimiter
4: Read input string from user and store it in input_string
5: Read delimiter from user and store it in delimiter
6: string_len ← Length of input_string
7: delimiter_len ← Length of delimiter
8: search_len ← string_len − delimiter_len
9: function ISCORRECT(i, delimiter_len)
10:     for k ← 0 to delimiter_len − 1 do
11:         if input_string[i + k] ≠ delimiter[k] then
12:             return false
13:         end if
14:     end for
15:     return true
16: end function
17: for i ← 0 to search_len − 1 do
18:     if ISCORRECT(i, delimiter_len) then
19:         for k ← 0 to delimiter_len − 1 do
20:             check[i + k] ← true
21:         end for
22:     end if
23: end for
24: sequence_flag ← true
25: Print "Separated tokens :"
26: for i ← 0 to string_len − 1 do
27:     if check[i] is false then
28:         if not sequence_flag then
29:             Print new line
30:             Print input_string[i]
31:             sequence_flag ← true
32:         else
33:             Print input_string[i]
34:         end if
35:     else
36:         sequence_flag ← false
37:     end if
38: end for
```

결과 화면



예시와 같은 입력을 넣어 같은 결과를 얻었다.

고찰

처음에 구분자 조건을 보지 않고 구분자를 하나의 char형으로만 보고 구현하여 낭패를 보았다.
구분자 인덱스를 가면 개행하고 넘기는 식으로 구현했는데 이후 구분자 길이 조건을 보고 다시 코드를 만들었다.
구분자 배열을 입력받고 처리하는 부분이 헷갈려서 고민 중 더 최적화된 알고리즘이나 방법은 있었겠지만 따로 check 배열을 만들어 구분하게 되었다.
check배열과 flag를 활용하여 0, 1의 경우의 수대로 각각의 경우에 따라 명령을 나누어서 해결하였다.

5.

문제 설명

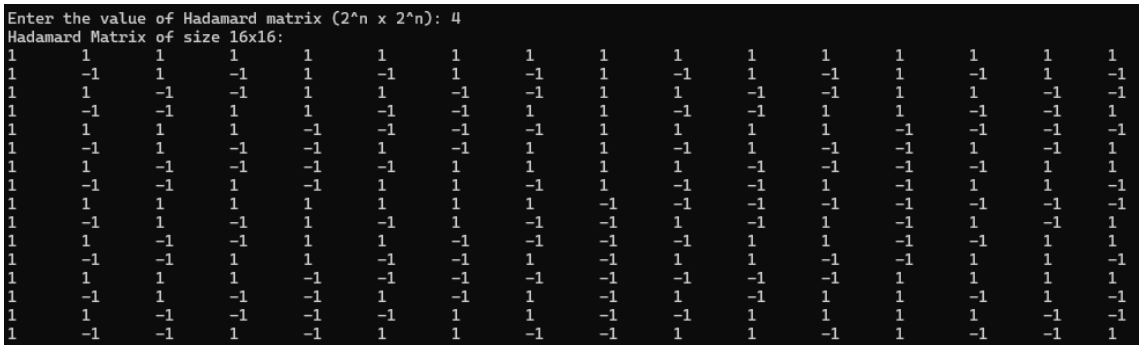
지수 승 n을 입력받아 재귀를 활용한 아다마르 행렬을 구현하는 프로그램을 작성하는 문제이다.

알고리즘

Algorithm 1 Hadamard Matrix Generation

```
function HadamardMat(rows, cols, power_n, invert_minus)
    if power_n = 1 then
        if invert_minus = false then
            mat[rows][cols] ← 1
        else
            mat[rows][cols] ← −1
        end if
    else
        power_n ← power_n/2
        if invert_minus = true then
            HadamardMat(rows, cols, power_n, 1)
            HadamardMat(rows, cols + power_n, power_n, 1)
            HadamardMat(rows + power_n, cols, power_n, 1)
            HadamardMat(rows + power_n, cols + power_n, power_n, 0)
        else
            HadamardMat(rows, cols, power_n, 0)
            HadamardMat(rows, cols + power_n, power_n, 0)
            HadamardMat(rows + power_n, cols, power_n, 0)
            HadamardMat(rows + power_n, cols + power_n, power_n, 1)
        end if
    end if
end function
Declare mat as a 2D array                ▷ Pointer to a pointer to an integer
Read integer n from user                 ▷ Size of Hadamard matrix
power_n ← 1
for i ← 0 to n − 1 do
    power_n ← power_n * 2
end for
Allocate memory for mat as power_n × power_n array of integers
HadamardMat(0, 0, power_n, 0)
Print "Hadamard matrix of size" power_n × power_n
for i ← 0 to power_n − 1 do
    for j ← 0 to power_n − 1 do
        Print mat[i][j]
    end for
    Print new line
end for
Deallocate memory for mat
```

결과 화면



4를 입력하여 2^4=16, 16*16 크기의 아다마르 행렬이 생성되었다.

고찰

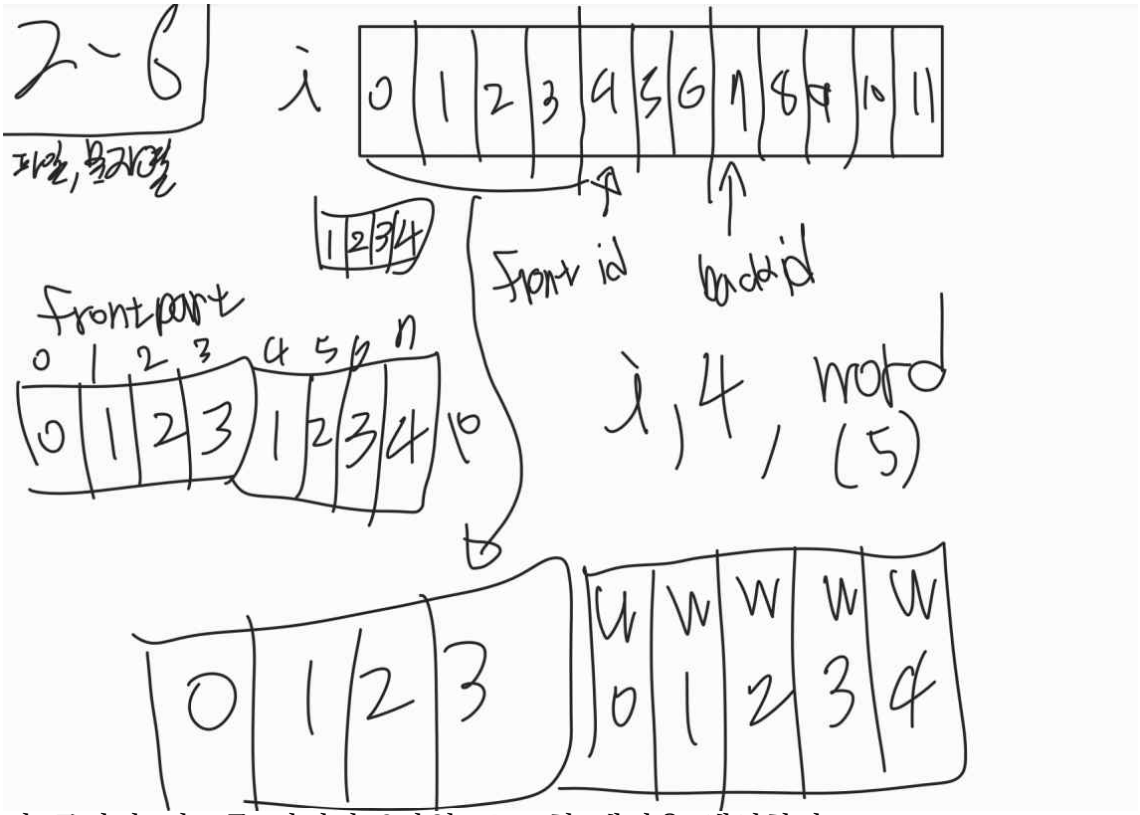
재귀를 어떻게 해야할 지 헷갈려 처음엔 가장 작은 정사각형의 호출만 되었다. 나머지 부분이 쓰레기값이 출력되거나 이상한 인덱스를 참조하여 오류가 생기기도 하였다.
문제인 부분은 인덱스를 어떻게 넘기느냐였고 이 부분은 이중 포인터를 전역으로 두면서 함수의 인수에서 처리하는 것을 줄이고 인덱스를 집중적으로 보아 해결할 수 있었다.

6.

문제 설명

6번 문제는 파일 열기, 단어 찾기, 단어 변환, 단어 인덱스 입력받고 삽입,단어 삭제, 저장, 종료 커맨드를 실행하는 프로그램이다.
이때, 예시에서 사용한 파일은 제공된 txt 파일이다.

알고리즘



각 줄마다 정보를 읽어와 2차원 char형 배열을 생성한다.
삽입,변환 시 인덱스를 구하여 앞, 뒤 배열을 다른 배열에 저장하고, 삽입될 단어를 앞-단어-뒤 배열 순으로 기존 행에 크기를 해제, 재할당 후 복사하여 저장한다.
단어 탐색은 4번과 같이 한 줄에서 단어를 한 인덱스씩 비교하고 이동하면서 true가 반환되면 각 함수에 따른 명령이 실행된다.
배열에 인덱스 저장 또는 삽입 명령 실행 또는 변환 명령이 실행된다.
아래는 main 함수의 커맨드 별 입력을 제외한 함수들의 수도코드이다.

Algorithm 1 my_strcmp

```
1: function MY_STRCMP(input, command)
2:     for i ← 0 to length_of_command do
3:         if input[i] > command[i] then
4:             return 1
5:         else if input[i] < command[i] then
6:             return −1
7:         end if
8:     end for
9:     return 0
10: end function
```

Algorithm 2 my_strlen

```
1: function MY_STRLEN(arr)
2:     len ← 0
3:     for i ← 0 to 999 do
4:         if arr[i] = n0 then
5:             return len
6:         end if
7:         len ← len + 1
8:     end for
9:     return len
10: end function
```

Algorithm 3 my_strcpy

```
1: function MY_STRCPY(arr1, arr2, arr2_len)
2:     for i ← 0 to arr2_len − 1 do
3:         arr1[i] ← arr2[i]
4:     end for
5:     arr1[arr2_len] ← n0
6: end function
```

Algorithm 4 my_strcat

```
1: function MY_STRCAT(arr1, arr2)
2:     i ← 0
3:     while arr1[i] ≠ n0 do
4:         i ← i + 1
5:     end while
6:     j ← 0
7:     while arr2[j] ≠ n0 do
8:         arr1[i + j] ← arr2[j]
9:         if arr2[j + 1] = n0 then
10:            arr1[i + j + 1] ← n0
11:        end if
12:        j ← j + 1
13:    end while
14: end function
```

Algorithm 6 search_word (search command)

```
1: function SEARCH_WORD(inputline, input_command2, count, rows_count)
2:     word_len ← MY_STRLEN(input_command2)
3:     line_len ← MY_STRLEN(inputline)
4:     search_len ← line_len − word_len
5:     for i ← 0 to search_len do
6:         found ← true
7:         for j ← 0 to word_len do
8:             if inputline[i + j] ≠ input_command2[j] then
9:                 found ← false
10:                break
11:            end if
12:        end for
13:        if found then
14:            rows[count] ← rows_count
15:            cols[count] ← i
16:            count ← count + 1
17:        end if
18:    end for
19: end function
```

Algorithm 7 search_word (delete)

```
1: function SEARCH_WORD(inputline, word)
2:     word_len ← MY_STRLEN(word)
3:     line_len ← MY_STRLEN(inputline)
4:     search_len ← line_len − word_len
5:     for i ← 0 to search_len do
6:         found ← true
7:         for j ← 0 to word_len do
8:             if inputline[i + j] ≠ word[j] then
9:                 found ← false
10:                break
11:            end if
12:        end for
13:        if found then
14:            DELETE_N_MERGE(inputline, word_len, i)
15:        end if
16:    end for
17: end function
```

Algorithm 8 search_word (change)

```
function SEARCH_WORD(inputline, word1, word2)
    word_len ← MY_STRLEN(word1)
    line_len ← MY_STRLEN(inputline)
    search_len ← line_len
 - word_len    for i ← 0 to search_len do
  found ← true
  for j ← 0 to word_len do
      if inputline[i + j] ≠ word1[j] then
          found ← false
          break
      end if
  end for
  if found then
      INSERT(inputline, word2, i, i + word_len)
  end if

end function
```

Algorithm 9 insert (change&insert)

```
function INSERT(line, word, front_id, back_id)
    MY_STRCPY(frontpart, line, front_id)
    MY_STRCAT(frontpart + front_id, word)
    front_len ← MY_STRLEN(frontpart)
    MY_STRCAT(frontpart + front_len, line + back_id)
    front_len ← MY_STRLEN(frontpart)
    delete[] line
    line ← nullptr
    line ← new char[front_len + 1]
    MY_STRCPY(line, frontpart, front_len)
    ARRAY_CLEAR(frontpart)
end function
```

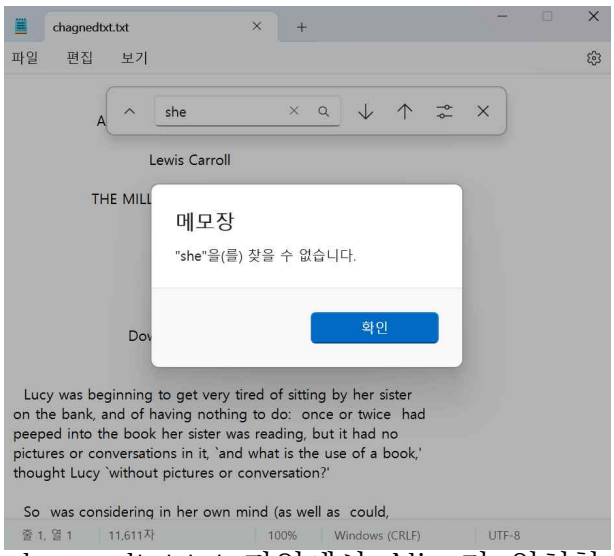
Algorithm 10 delete_N_merge (delete)

```
function DELETE_N_MERGE(line, word_len, cols)
    ARRAY_CLEAR(frontpart)
    ARRAY_CLEAR(backpart)
    MY_STRCPY(frontpart, line, cols)
    MY_STRCPY(backpart, line + cols + word_len, (my_strlen(line) − cols −
word_len + 1))
    MY_STRCAT(frontpart, backpart)
    line_len ← MY_STRLEN(frontpart)
    MY_STRCPY(line, frontpart, line_len)
    ARRAY_CLEAR(frontpart)
    ARRAY_CLEAR(backpart)
end function
```

결과 화면

```
open
./6.txt
search Alice
==='Alice' search(28)===
(15, 2), (19, 8), (27, 56), (33, 47), (40, 30), (44, 48), (61, 18), (71, 9), (76, 29), (92, 54), (98, 43), (109, 2), (113, 10), (121, 9), (127, 4), (135, 2), (141, 40)
, (145, 26), (152, 37), (157, 0), (169, 51), (181, 33), (189, 41), (196, 53), (205, 51), (213, 23), (220, 26), (229, 32)
--------------------
search Lucy
==='Lucy' search(0)===

--------------------
change Alice Lucy
===change===
Alice -> Lucy
--------------------
search Alice
==='Alice' search(0)===

--------------------
search Lucy
==='Lucy' search(28)===
(15, 2), (19, 8), (27, 56), (33, 47), (40, 30), (44, 48), (61, 18), (71, 9), (76, 29), (92, 54), (98, 43), (109, 2), (113, 10), (121, 9), (127, 4), (135, 2), (141, 40)
, (145, 26), (152, 37), (157, 0), (169, 51), (181, 33), (189, 41), (196, 53), (205, 51), (213, 23), (220, 26), (229, 32)
--------------------
search she
==='she' search(80)===
(16, 57), (21, 5), (21, 53), (29, 54), (30, 44), (34, 20), (34, 46), (36, 44), (41, 29), (45, 46), (48, 36), (48, 62), (49, 22), (50, 46), (51, 23), (52, 19), (53, 54)
, (54, 15), (55, 32), (55, 43), (57, 12), (58, 60), (68, 49), (80, 12), (83, 10), (86, 55), (87, 14), (89, 24), (101, 33), (102, 42), (103, 5), (103, 58), (106, 40), (
109, 31), (110, 9), (115, 45), (116, 50), (122, 26), (123, 14), (125, 11), (130, 42), (131, 8), (132, 33), (136, 43), (138, 4), (140, 0), (148, 62), (149, 36), (151, 2
3), (158, 8), (159, 4), (164, 56), (172, 41), (172, 59), (184, 25), (185, 43), (187, 16), (187, 55), (188, 30), (191, 17), (192, 40), (195, 53), (197, 5), (197, 26), (
197, 36), (198, 28), (199, 0), (199, 10), (199, 44), (200, 37), (202, 9), (207, 53), (208, 35), (209, 51), (211, 11), (218, 12), (227, 35), (228, 10), (234, 5), (234,
40)
--------------------
delete she
===delete===
Delete she
--------------------
search she
==='she' search(0)===

--------------------
save chagnedtxt.txt
===save===
Save the file as "chagnedtxt.txt"
exit
Exit the program
```

파일 이름을 6.txt로 바꾸어 파일을 open 하였다.
파일에서 Alice를 검색하여 인덱스를 출력하였다.
Lucy는 없었다.
Alice를 Lucy로 바꾸었다.
이후 Alice는 없어졌고 Lucy는 기존의 Alice의 개수였던 28개가 되었다.
이후 she 단어를 검색하여 80개를 얻었다.
delete she 후 다시 검색해보니 she가 사라졌다.
이후 저장 후 종료하였다.



changedtxt.txt 파일에서 Alice가 위치한 자리는 Lucy로 바뀌어 있고, she는 찾을 수 없었다. 예상 결과가 잘 저장된 모습이다.

7.

문제 설명

7번 문제는 산술 연산과 괄호를 지원하는 중위표기식 연산 프로그램이다.
이때, 스택을 이용해 문제를 해결해야 한다.
하지만, 스택 라이브러리를 사용하지 않고 스택을 구현하여 사용해야 한다.

알고리즘

1. 중위 표기식을 후위 표기식으로 변환
2. 후위 표기식을 연산

1: 중위 표기 -> 후위 표기 ::배열에 저장
1-1: 숫자는 바로 저장
1-1-2: 숫자는 나오면 다음, 다음다음 인덱스 조사 후 함께 저장 후 인덱스 이동
1-2: 스택이 비어있으면 연산자를 스택에 저장
1-3: 스택의 top<들어갈 연산자의 순서면 연산자를 스택에 저장
1-4: 1-3이 아니면 pop-저장하다가 2,3이 될 때까지, 이후 저장
1-5: 수식 끝나면 전부 pop, 배열에 저장
1-6: 우선 순위는 사칙연산과 동일
1-7: 여는 괄호는 스택에 저장
1-8: 여는 괄호 다음 연산자는 스택에 바로 추가
1-9: 닫는 괄호 나오면 여는 괄호 나올 때까지 pop-저장

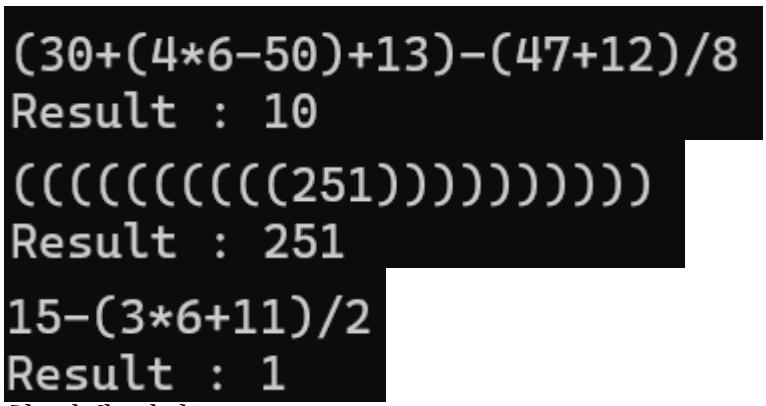
2: 후위 표기 -> 연산
2-1: 숫자는 스택에 저장
2-2: 연산자가 나오면 두 개의 숫자를 스택에서 꺼내 연산 후 스택에 저장
2-3: 배열이 끝나면 스택에 남은 하나의 숫자를 출력

3. 출력
결과가 0보다 크면 0.5 더해 출력
결과가 0보다 작으면 0.5를 빼 출력

Algorithm 1 Infix to Postfix Conversion and Evaluation

```
1: Function Main()
2: a ← CreateStack(Operators stack)
3: input ← ReadInput(Infix expression)
4: b ← CreateStack(Numbers stack)
5: reward ← Create2DArray(Array to store postfix expression)
6: oper ← {'+', '-', '*', '/', '(', ')'}          ▷ Allowed operators and parentheses
7: reward_size ← 0                                ▷ Size of the postfix expression array
8: for i ← 0 to LengthOf(input) do
9:     if input[i] is a number then
10:        Append(reward[reward_size], input[i])          ▷ Append single-digit number
11:        if input[i + 1] is also a number then
12:            Append(reward[reward_size], input[i + 1])          ▷ Append double-digit number
13:            i ← i + 1
14:        end if
15:        if input[i + 1] is also a number then
16:            Append(reward[reward_size], input[i + 1])          ▷ Append triple-digit number
17:            i ← i + 1
18:        end if
19:        reward_size ← reward_size + 1
20:    else
21:        if a is empty or input[i] is '(' or (a's top is '(' and input[i] is not ')) then        Push(a,input[i])
           if input[i] is ')' then
               while a's top is not '(' do
                   Append(reward[reward_size], a's top)
                   Pop(a)
                   reward_size ← reward_size + 1
               end while
               Pop(a)                                                          ▷ Pop '('
           else if input[i] is '*' or '/' then
               if a's top is '+' or '- then                    Push(a,input[i])
                   if
                       while   dothena's top is '*' or '/'
                           Append(reward[reward_size], a's top)
                           Pop(a)
                           reward_size ← reward_size + 1
                       end while
                       Push(a, input[i])
                   end if
               else if input[i] is '+' or '-' then
                   while a's top is not '(' do
                       Append(reward[reward_size], a's top)
                       Pop(a)
                       reward_size ← reward_size + 1
                   end while
                   Push(a, input[i])
               end if
           end if
       end for
       while a is not empty do
           Append(reward[reward_size], a's top)
           Pop(a)
           reward_size ← reward_size + 1
       end while
       i ← 0
       while reward[i] is not empty do
```

결과 화면



첫 번째 결과는
30+(24-50)+13-(59/8)
=54-50+13-7.375
=17-7.375
=9.625==>10

두 번째 결과는 문제 조건에서 벗어났지만 괄호쌍 10개를 입력받았을 때 출력할 수 있었고, 251로 제대로 결과가 출력되었다.

세 번째 결과는 예시의 경우에서 2가 6으로 바뀐 경우로,
15-(29)/2
=15-14.5
=0.5==>1

고찰

구현하면서 스택 중에서 배열에 숫자와 연산자를 어떻게 구분해야 할지,
스택 구현 자체의 문제도 있었고 숫자가 여러 자리에 걸쳐 있으면 어떻게 받아야 할지 고민하였다.
atoi 구현, char 2차원 배열로 후위 표기식 저장, 후위 표기 변환 알고리즘을 규칙을 정해 구현하여 해결할 수 있었다.

8.

문제 설명

8번 문제는 사람, 학번, 출석 점수, 과제 점수, 시험 점수를 입력받고
삽입, 출력, 점수 변환, 출력, 종료 커맨드를 지원하는 프로그램을 만드는 것이다.
출석 10%, 과제 40%, 시험점수 50%로 최종 점수를 계산한다.

알고리즘

Algorithm 1 Student and School Management Program

```
Input:
    command: User input command
    input_name, input_id: Student name and ID
    input_aScore, input_eScore, input_att: Assignment, exam, and attendance scores
Output:
    Student information or error message

procedure MAIN
    a ← New school object
    command_list ← ["insert", "find", "change", "print", "exit"]

    while true do
        command ← User input
        if command = "insert" then                          ▷ Add a new student
            input_name, input_id, input_aScore, input_eScore, input_att ← User input
            NEW_STUDENT(a, input_name, input_id, input_aScore, input_eScore, input_att)
        else if command = "find" then     ▷ Search for student information
            input_name ← User input
            PRINT_FIND(a, input_name)
        else if command = "change" then    ▷ Change student information
            input_name, input_aScore, input_eScore, input_att ← User input
            CHANGE_SCORES(a, input_name, input_aScore, input_eScore, input_att)
        else if command = "print" then     ▷ Print all student information
            PRINT_ALL(a)
        else if command = "exit" then                       ▷ Exit the program
            break
        else
            print "Invalid command."
        end if
    end while
    print "Program terminated"
end procedure

procedure NEW_STUDENT(a, name, id, aScore, eScore, att)
    Create a new student object with given parameters
    Add the student object to the school object a
end procedure

procedure PRINT_ALL(a)
    Retrieve all student information from school object a
    Print all student information
end procedure

procedure PRINT_FIND(a, name)
    Search for student information by name in school object a
    Print student information if found, otherwise print error message
end procedure

procedure CHANGE_SCORES(a, name, aScore, eScore, att)
    Search for student information by name in school object a
    Update student's scores with new values
end procedure
```

[1]

결과 화면

```
insert a1 2023000001 90 90 90
insert a2 2023000002 80 70 100
insert a3 2023000003 90 90 90
print
=====print=====
Name : a1
Student ID : 2023000001
Final Score : 90
---------------
Name : a2
Student ID : 2023000002
Final Score : 77
---------------
Name : a3
Student ID : 2023000003
Final Score : 90
---------------
find a1
=====find=====
Name : a1
Student ID : 2023000001
Final Score : 90
---------------
change a2 90 90 100
print
=====print=====
Name : a1
Student ID : 2023000001
Final Score : 90
---------------
Name : a2
Student ID : 2023000002
Final Score : 91
---------------
Name : a3
Student ID : 2023000003
Final Score : 90
---------------
exit
Exit the program
```

```
//////////////////////
insert a1 2023000001 90 90 90
insert a2 2023000002 80 70 100
insert a3 2023000003 90 90 90
print
find a1
change a2 90 90 100
print
exit
//////////////////////
```

위 커맨드를 입력했을 때 결과가 위와 같이 나왔다.
문제에서 제시한 결과와 같다.

고찰

문제에서 클래스가 처음 등장하여 클래스를 활용하면서 클래스의 private과 public의 구분과 그 활용을 알 수 있었다.
또한, class의 메서드를 구현하여 main문의 길이를 줄이면서 보다 깔끔한 형태의 코드를 작성할 수 있었다고 느꼈다.

9.

문제 설명

9번 문제는 두 개의 클래스, 한 클래스 안에 다른 클래스가 들어가있는 이중 클래스를 구현하는 문제이다.
이때, 8번 문제와 비슷하게 학생을 관리하는 프로그램이고, A학점을 받는 학생, B를 받는 학생, 정렬 커맨드가 추가되어 구현해야 한다.

알고리즘

Algorithm 1 Student and School Management Program

```
Input:
    command: User input command
    input_name, input_id: Student name and ID
    input_score: Student score
Output:
    Student information or error message

procedure MAIN
    a ← New school object
    command_list ← ["new_student", "sort_by_score", "print_all", "print_A_grade", "print_B_grade", "exit"]

    while true do
        input_command ← User input
        if input_command = "new_student" then ▷ Add a new student
            input_name, input_id, input_score ← User input
            NEW_STUDENT(a, input_name, input_id, input_score)
        else if input_command = "sort_by_score" then ▷ Sort students by score
            SORT_BY_SCORE(a)
        else if input_command = "print_all" then ▷ Print all student information
            PRINT_ALL(a)
        else if input_command = "print_A_grade" then ▷ Print students with A grade
            PRINT_A_GRADE(a)
        else if input_command = "print_B_grade" then ▷ Print students with B grade
            PRINT_B_GRADE(a)
        else if input_command = "exit" then ▷ Exit the program
            break
        else
            print "Error: Invalid command"
        end if
    end while
    print "Program terminated"
end procedure

procedure NEW_STUDENT(a, name, id, score)
    Create a new student object with given name, id, and score
    Add the student to the school object a
end procedure

procedure SORT_BY_SCORE(a)
    Sort students in school object a by score
end procedure

procedure PRINT_ALL(a)
    Print all student information in school object a
end procedure

procedure PRINT_A_GRADE(a)
    Print students with A grade in school object a
end procedure

procedure PRINT_B_GRADE(a)
    Print students with B grade in school object a
end procedure
```

결과 화면

```
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
=====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
---------------
sort_by_score
print_all
=====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
---------------
print_A_grade
=====A grade=====
Name : Olivia
Student ID : 2013000005
Score : 95
---------------
print_B_grade
=====B grade=====
Name : Sophia
Student ID : 2013000004
Score : 80
---------------
exit
Exit the program
```

예시에 나온 입력을 입력하였을 때 위와 같은 결과를 얻었다.
예상과 같은 결과가 출력되었다.

고찰

9번 문제는 8번과 비슷하였지만
헷갈렸던 부분은 school 클래스에 student * 형식 변수 하나를 바꾸지 않고 어떻게 관리하라는 것인지 헷갈렸다.
각 칸에 할당할 수 있도록 포인터 배열이거나
링크드 리스트로 사용할 수 있도록 포인터가 따로 있으면 가능하지만 저런 형식을 가진 상태에서 어떻게 구현할지 몰라 동적 배열을 할당하고 크기가 넘어가면 다시 크기를 늘리고 삭제, 재할당하여 크기를 늘리도록 만들었다.
다시 만든다면 더욱 고민하여 링크드 리스트로 만들 수도 있을 것이란 생각이 들었다.

10.

문제 설명

10번 문제는 txt 파일에서 학생 정보를 받아오고 9번의 커맨드를 구현하는 프로그램이다. 9번의 new_student 명령을 load_student 명령으로 바꾼다.

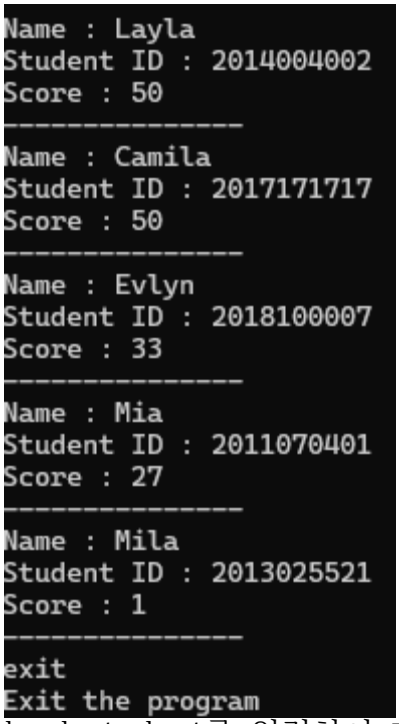
알고리즘

Algorithm 1 Student and School Management Program

```
Input:
    command: User input command
    input_name, input_id: Student name and ID
    input_score: Student score
Output:
    Student information or error message

procedure MAIN
    a ← New school object
    command_list ← ["load_student", "sort_by_score", "print_all", "print_A_grade", "print_B_grade", "exit"]
    read_file ← File reader object

    while true do
        input_command ← User input
        if input_command = "load_student" then ▷ Load student information from file
            read_file.open("student.txt")
            while not end of file do
                inputline ← Read next line from file
                TOKENIZER(a, inputline) ▷ Tokenize and add student to school
            end while
            read_file.close()
        else if input_command = "sort_by_score" then ▷ Sort students by score
            SORT_BY_SCORE(a)
        else if input_command = "print_all" then ▷ Print all student information
            PRINT(a)
        else if input_command = "print_A_grade" then ▷ Print students with A grade
            PRINT_A_GRADE(a)
        else if input_command = "print_B_grade" then ▷ Print students with B grade
            PRINT_B_GRADE(a)
        else if input_command = "exit" then ▷ Exit the program
            break
        else
            print "Error: Invalid command"
        end if
    end while
    print "Program terminated"
end procedure

procedure TOKENIZER(a, line)
    line_len ← Length of line
    name ← Empty string
    stu_id ← Empty string
    score ← 0
    tag ← 0
    flag ← 0

    for each character in line do
        if character is comma or end of line then
            switch tag
            case 0: MY_STRCPY(name, line, i) ▷ Copy name
            case 1: MY_STRCPY(stu_id, line + flag, i - flag) ▷ Copy student ID
```

결과 화면

```
load_student
print_all
=====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
---------------
Name : Charlotte
Student ID : 2019101010
Score : 51
---------------
Name : Isabella
Student ID : 2017905301
Score : 79
---------------
Name : Mia
Student ID : 2011070401
Score : 27
---------------
Name : Luna
Student ID : 2015042005
Score : 90
---------------
Name : Harper
Student ID : 2013402139
Score : 64
---------------
Name : Gianna
Student ID : 2018000007
Score : 81
---------------
Name : Evlyn
Student ID : 2018100007
Score : 33
---------------
Name : Aria
Student ID : 2018001007
Score : 50
---------------
Name : Ella
Student ID : 2015050505
Score : 77
---------------
Name : Ellie
Student ID : 2016052317
Score : 85
---------------
Name : Mila
Student ID : 2013025521
Score : 1
---------------
Name : Layla
Student ID : 2014004002
Score : 50
---------------
Name : Avery
Student ID : 2013002178
Score : 90
---------------
Name : Camila
Student ID : 2017171717
Score : 50
---------------
Name : Lily
Student ID : 2018181818
Score : 80
---------------
sort_by_score
print_all
=====print=====
Name : Olivia
Student ID : 2013000005
Score : 95
---------------
Name : Luna
Student ID : 2015042005
Score : 90
---------------
Name : Avery
Student ID : 2013002178
Score : 90
---------------
Name : Ellie
Student ID : 2016052317
Score : 85
---------------
Name : Gianna
Student ID : 2018000007
Score : 81
---------------
Name : Sophia
Student ID : 2013000004
Score : 80
---------------
Name : Lily
Student ID : 2018181818
Score : 80
---------------
Name : Isabella
Student ID : 2017905301
Score : 79
---------------
Name : Ella
Student ID : 2015050505
Score : 77
---------------
Name : Amelia
Student ID : 2014100001
Score : 71
---------------
Name : Ava
Student ID : 2011020306
Score : 64
---------------
Name : Harper
Student ID : 2013402139
Score : 64
---------------
Name : Emma
Student ID : 2013020103
Score : 55
---------------
Name : Charlotte
Student ID : 2019101010
Score : 51
---------------
Name : Aria
Student ID : 2018001007
Score : 50
---------------
Name : Layla
Student ID : 2014004002
```



load_student를 입력하여 파일에서 불러오고,
print_all
sort_by_score
print_all
exit
커맨드를 입력하여 위와 같은 결과를 얻었다.
이는 예상 결과와 같다.

고찰

10번은 9번의 코드를 활용하여 만들었기 때문에 파일 입출력 부분 외엔 같다. 파일 입출력에서 txt 파일에 , 구분자를 기준으로 띄어쓰기가 2개인 경우, 숫자 뒤에 띄어쓰기가 있는 경우가 있어 delete_space 함수를 이용해 공백을 모두 지우고 각 포맷을 입력받아 학생의 정보로 저장하였다.

11.

문제 설명

11번 문제는 연결 리스트 구현하여 숫자를 입력받을 때, 삭제할 때마다 연결 리스트를 출력하고, exit 커맨드로 종료하는 프로그램이다.

알고리즘

1. cmp함수는 이전 코드와 동일하다.
2. main 함수는 커맨드를 입력받고 class의 메서드를 실행시킨다.
3. 노드 class는 내부 private 변수를 반환,세팅하는 함수와 생성,소멸자로 이루어져있다.
4. list 함수는 헤드,테일 포인터 변수를 private으로 가지고 있다.

4-1: 생성, 소멸자는 각각의 포인터로 연결 리스트를 초기화, 삭제 처리를 한다.

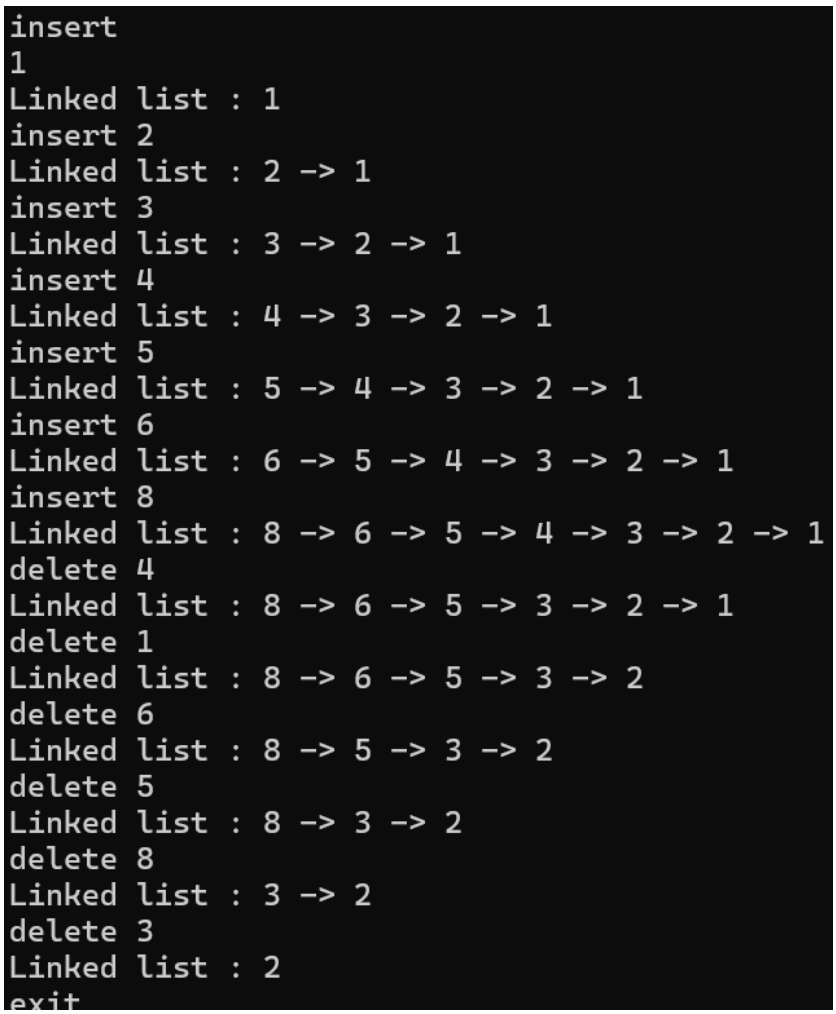
4-2: setnode

setnode는 하나의 노드를 생성하고 헤드 위치에 노드를 삽입한다.
이후 노드의 헤드는 생성한 노드가 되고 노드는 이전 노드를 가리켜 연결된다.

4-3: del

del함수는 제일 앞의 head부터 tail까지 노드를 순차적으로 돌면서 삭제값을 입력받아 같은 숫자가 있을 시 삭제한다.
print함수는 head부터 tail까지 숫자를 출력 포맷에 맞춰 출력한다.

결과 화면



입력에 따라 입력이 머리에 들어가고, 삭제시 앞에 있는 하나의 노드가 삭제된다.
exit를 입력하여 프로그램이 종료된다.

고찰

클래스로 링크드 리스트를 만드는 것이 익숙치 않아 처음 힘든 부분이 있었다. 하지만 1차원이기에 연결하는 경우의 수를 나누어 한가지씩 조건 제어를 통해 연결 리스트를 제작하였다.
소멸자에서 아무것도 존재하지 않을 시 head 포인터가 null을 반환하는 문제가 중간에 발생하여 이를 처리하는 조건문을 넣어 해결하였다.

12.

문제 설명

12번 문제는 데이터 파일에서 한 단어씩 데이터를 가져와 2차원 연결 리스트를 만드는 것이다.

알고리즘

Algorithm 1 SetNode Method

```
1:  function SetNode(name)
2:      new_node ← Node(name)
3:      if head = NULL then
4:          head ← new_node
5:      else
6:          tmp ← head
7:          pre ← NULL
8:          while true do
9:              initial_diff ← GetInitial(new_node) − GetInitial(tmp)
10:             if initial_diff < 0 then
11:                 SetDown(new_node, tmp)
12:                 if tmp = head then
13:                     head ← new_node
14:                 else
15:                     SetDown(pre, new_node)
16:                 end if
17:                 break
18:             else if initial_diff > 0 then
19:                 if GetDown(tmp) = NULL then
20:                     SetDown(tmp, new_node)
21:                     break
22:                 else
23:                     pre ← tmp
24:                     tmp ← GetDown(tmp)
25:                 end if
26:             else
27:                 while true do
28:                     name_diff ← StringCompare(GetName(new_node), GetName(tmp))
29:                     if name_diff < 0 then
30:                         if pre = NULL then
31:                             SetNext(new_node, tmp)
32:                             head ← new_node
33:                         else if GetInitial(pre) ≠ GetInitial(tmp) then
34:                             SetNext(new_node, tmp)
35:                             if GetDown(tmp) ≠ NULL then
36:                                 SetDown(new_node, GetDown(tmp))
37:                             end if
38:                             SetDown(pre, new_node)
39:                         else
40:                             SetNext(new_node, tmp)
41:                             SetNext(pre, new_node)
42:                         end if
43:                         break
44:                     else if name_diff > 0 then
45:                         if GetNext(tmp) = NULL then
46:                             SetNext(tmp, new_node)
47:                             break
48:                         else
49:                             pre ← tmp
50:                             tmp ← GetNext(tmp)
51:                         end if
52:                     end if
53:                 end while
54:                 break
55:             end if
56:         end while
57:     end if
58: end function
```

my_string 함수들은 이전 코드와 동일하다.

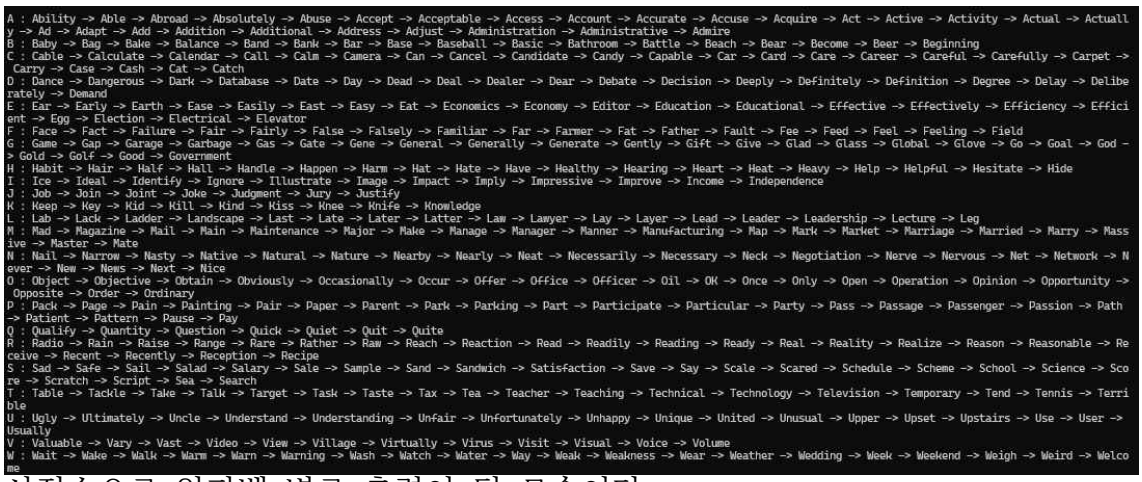
생성자와 소멸자는 각각 포인터의 초기화와 리스트의 삭제 역할이다.

print함수는 리스트 전체를 출력한다.

get함수들은 getter로, private값을 반환한다.

main함수는 파일에서 한줄씩 불러오고 메서드를 실행시킨다.

결과화면



사전순으로 알파벳 별로 출력이 된 모습이다.
2차원으로 구현되어 각 알파벳별로 출력, 단어별로 출력하였다.

고찰

1학년 고급 C 프로그래밍의 프로젝트 대체 과제로 출제된 문제와 비슷한 결이 있었다. 그 당시의 문제는 다른 데이터를 비교하여 2차원의 주 리스트, 부 리스트로 나눠지는 부분이었다.
이번 문제도 비슷하게 알파벳 이니셜로 주 리스트를 설정, 단어 사전 순으로 부 리스트를 설정하여 위와 같은 파일에서 데이터 받아오기, 출력을 실행하게 되었다. 노드 삽입 시 여러 조건이 있어 각각의 조건과 경우의 수를 비교하여 위에서의 설명과 같이 구현하였다.