Object-Oriented Programming Report

Assignment 2-3

Professor	Donggyu Sim
Department	Computer engineering
Student ID	2020202031

Name	Jaehyun Kim
Class (Design / Laboratory)	1 / B
Submission Date	2023. 04. 28

Program 1

❑ 문제 설명

Employee class 를 정의하고, 사용자로부터 입력 받은 명령어에 맞게 Employee 객체들을 생성하거나 적절히 수정한다. 또한 명령어에 따라서 객체를 전체 출력하거나 일부 골라서 출력하는 것 또한 가능하도록 코드를 작성한다.

main 에서 while 문을 돌면서 사용자로부터 명령어를 입력 받고, 입력 받은 명령어를 strcmp 함수를 통해 어떤 명령어인지 판별하고 그에 따른 코드를 실행합니다.

매개변수가 있는 생성자와 복사 생성자는 깊은 복사를 진행하도록 했고, 소멸자에선 깊은 복사를 통해 동적 할당된 메모리를 해제해줬습니다. change 메소드에서는 원래 있던 동적 메모리들을 해제시키고 새로운 메모리를 동적 할당하는 식으로 구현했습니다.

❑ 결과 화면

```
insert John 23 USA designer
insert Jaehyun 23 Korea student
insert Neymar 30 Brasil player
insert James 28 UK developer
insert Nacamura 25 Japan manager
print
=====print=====
Name: John
Age: 23
Country: USA
Job: designer
----------
Name: Jaehyun
Age: 23
Country: Korea
Job: student
----------
Name: Neymar
Age: 30
Country: Brasil
Job: player
----------
Name: James
Age: 28
Country: UK
Job: developer
----------
Name: Nacamura
Age: 25
Country: Japan
Job: manager
----------
find Jaehyun
=====find=====
Name: Jaehyun
Age: 23
Country: Korea
Job: student
----------
```

5 명의 Employee 를 insert 하고 print 와 find 를 한 결과입니다.
출력이 올바르게 작동하는 것을 확인할 수 있습니다

```
insert Jessica 23 USA dancer
insert xi 48 China firefighter
insert shasha 40 Germany salesman
insert Jeongkuk 29 Korea i-dol
insert Messi 38 Argentina player2
print
=====print=====
Name: John
Age: 23
Country: USA
Job: designer
----------
Name: Jaehyun
Age: 23
Country: Korea
Job: student
----------
Name: Neymar
Age: 30
Country: Brasil
Job: player
----------
Name: James
Age: 28
Country: UK
Job: developer
----------
Name: Nacamura
Age: 25
Country: Japan
Job: manager
----------
Name: Jessica
Age: 23
Country: USA
Job: dancer
----------
Name: xi
Age: 48
Country: China
Job: firefighter
----------
Name: shasha
Age: 40
Country: Germany
Job: salesman
----------
Name: Jeongkuk
Age: 29
Country: Korea
Job: i-dol
----------
Name: Messi
Age: 38
Country: Argentina
Job: player2
----------
```

추가로 5 명의 Employee 를 더 추가하고 print 했더니 10 명의 Employee 가 잘 출력되는 것을 확인할 수 있습니다.

```
insert HaHa 35 Korea singer
already 10 employees exist
find HaHa
=====find=====
No one has a name like HaHa
change Messi Mbappe 27 France player3
change HaHa Jongkuk 34 Korea singer
No one has a name likeHaHa
exit
```

Employee 를 10 명 채운 후 Employee 를 더 추가하려고 하면 "already 10 employees exist"라는 문구가 출력되고 추가명령이 실행되지 않습니다. 그렇기 때문에 find 명령어로 해당 Employee 를 찾으려고 해도 "No one has a name like ~~"라는 문구가 출력됩니다. change 명령어로 존재하지 않는 Employee 를 찾을 경우에도 "No one has a name like ~"라는 문구만이 실행됩니다. exit 을 입력하니 프로그램이 종료되는 것을 확인할 수 있습니다. 마지막 출력에서 like 와 name 사이에 공백이 없게 출력이 된 것을 확인하고 공백을 출력하도록 코드를 추가했습니다.

❑ 고찰

class 배열을 선언하면, 선언과 동시에 배열의 모든 객체들이 Default constructor 를 자동으로 실행해버려서 원하는 constructor 를 실행할 수 없었습니다. 그래서 class 포인터 배열을 선언해서 insert 명령어가 입력되면 0 번째 인덱스부터 동적할당을 통해 객체를 생성했고 이를 통해 매개변수가 있는 constructor 를 실행시켜 원하는 정보를 객체에 저장할 수 있었습니다.

Program 2

❑ 문제 설명

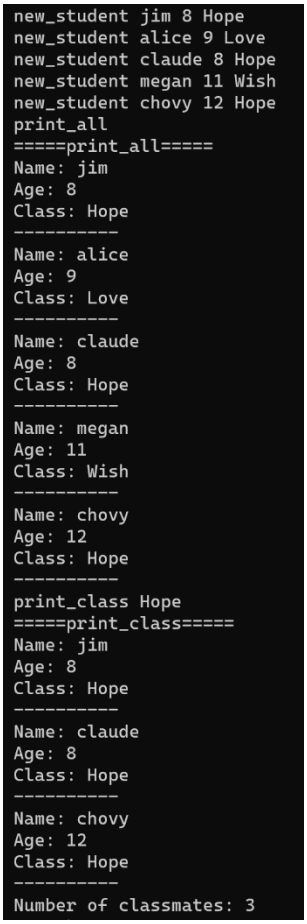
1 번 문제와 유사하게 사용자로부터 입력 받은 명령어에 따라 class Student 객체를 클래스 포인터 배열에 할당하고, 출력하는 등의 기능을 구현하면 되는데, 클래스 포인터 배열 선언부터, 이러한 기능들을 전부 School class 내부에서 관리하는 프로그램을 작성하는 문제입니다.

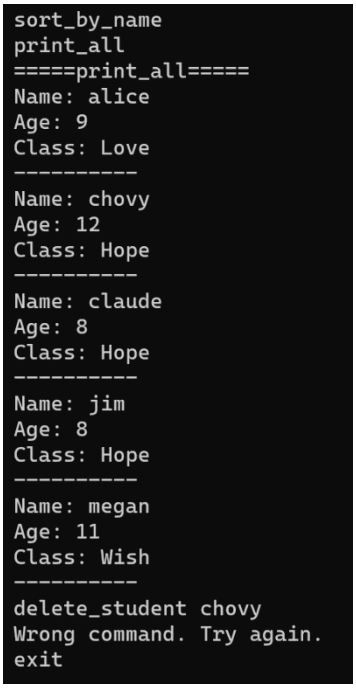
main 에서 while 문을 돌면서 사용자로부터 명령어를 입력 받고, 입력 받은 명령어를 strcmp 함수를 통해 어떤 명령어인지 판별하고 그에 따른 코드를 실행합니다.

매개변수가 있는 생성자가 깊은 복사를 진행하도록 했고, 소멸자에선 깊은 복사를 통해 동적 할당된 메모리를 해제해줬습니다.

Sort_by_name 메소드에서는 Bubble Sort 개념을 사용했습니다. strcmp 함수를 통해 두 객체의 name 아스키코드 값을 비교하여 전자가 더 클 경우 객체의 주소 값을 swap 했습니다.

❑ 결과 화면





학생 5 명을 추가하고 print_all 명령어를 입력하니 학생을 입력해준 순서대로 모두 출력되는 것을 확인할 수 있으며, print_class 로 특정 교실의 학생들만 출력하는 것

또한 확인할 수 있습니다. sort_by_name 으로 name 을 알파벳 오름차순으로 정렬한 것을 print_all 을 실행함으로써 확인할 수 있습니다. 또한, 존재하지 않는 명령어를 입력할 시 "Wrong command. Try again."이라는 문구가 출력되며 입력 대기 상태가 됩니다. exit 을 입력하면 프로그램이 종료됩니다.

❑ 고찰

sort_by_name command 를 구현하기 위해 School 의 메서드로 sort_by_name 을 선언했습니다. name 을 알파벳 오름차순으로 정렬하기 위해 Bubble Sort 개념을 사용하였습니다. 이때, 이중 for 문에서 범위 설정에 대한 실수를 반복적으로 범해 많은 시간을 빼앗겼습니다. 2 번 문제를 풀면서 Bubble Sort 에 대해 다시 한번 적응할 수 있는 시간이 된 것 같아 좋았습니다.

Program 3

❑ 문제 설명

to_string(float val)을 구현할 때, int 타입 변수에 float 타입 데이터를 저장하면 소수부분이 소실되고, 정수부분만 저장된다는 점을 이용하여 정수부분과 소수부분을 나누어 따로 길이를 구한 후 두 부분의 길이의 합 크기만큼 동적할당을 진행했습니다. 소수부분을 char 형으로 저장할 때도 소수부분에 10 을 곱하면 소수점 아래 첫번째 부분이 정수부분이 되고 이를 int 형 변수에 저장하면 소수점 아래 첫번째 부분을 int 형으로 다룰 수 있게 된다는 점을 활용했습니다.

❑ 결과 화면

```
string        : str1, str2
oopstd::string: strstr1, strstr2

==========c_str==========
str1   .c_str():   Hello
strstr1.c_str():   Hello

==========assign==========
str1   .assign("Hello World"):   Hello World
strstr1.assign("Hello World"):   Hello World

==========at==========
str1   .at(2):   l
strstr1.at(2):   l

==========clear==========
str1   .clear():
strstr1.clear():

==========push_back==========
str1   .pus_back('!'):   Hello World!
strstr1.pus_back('!'):   Hello World!

==========compare==========
str1   .compare(str2)    #str2 =     Hello World!:   0
strstr1.compare(strstr2) #str2str2 = Hello World!:   0

str1   .compare(str2)    #str2 =     Hello Hi!:   1
strstr1.compare(strstr2) #str2str2 = Hello Hi!:   1

str1   .compare(str2)     #str2   =   Hello Zoo!:   -1
strstr1.compare(&strstr2) #str2str2 = Hello Zoo!:   -1

==========replace==========
str1   .replace(5, 2, str2)     #str2  =   Hi:   HelloHiorld
strstr1.replace(5, 2, &strstr2) #strstr2 = Hi:   HelloHiorld

str1   .replace(5, 15, str2)     #str2  =   Hi:   HelloHi
strstr1.replace(5, 15, &strstr2) #strstr2 = Hi:   HelloHi
```

1) c_str: str1 과 strstr1 에 저장돼있는 문자열 "Hello"가 c_str()을 통해 반환되어 출력까지 완료된 것을 확인 가능합니다.
2) assign: "Hello"가 저장돼있던 str1 과 strstr1 에 "Hello World"를 assign 하고 출력했더니 "Hello World"가 출력되는 것으로 보아 assign 이 잘 실행됨을 확인 가능합니다.
3) at: "Hello World"가 저장돼있는 str1 과 strstr1 의 인덱스번호 2 번에는 'l'이 저장돼있고 'l'이 잘 출력되는 것을 확인 가능합니다.
4) clear: clear 를 통해 str1 과 strstr1 에 저장돼있던 "Hello World"문자열을 삭제하여 아무 것도 출력되지 않는 것을 확인 가능합니다.
5) push_back: 다시 str1 과 strstr1 에 "Hello world"를 assign 하고 push_back 으로 '!'를 붙여줘서 "Hello World!"가 출력되는 것을 확인 가능합니다.

6) compare: 비교 대상의 문자열 아스키코드 값이 완전히 동일하면 0, 크면 1, 작으면 -1 을 출력하는 것을 확인 가능합니다.

지금부터는 str1, strstr1 이 항상 "Hello World"입니다.

7) replace: 인자로 전달된 pose 가 문자열 범위 내에 존재한다면 잘 작동하지만 범위를 초과해버리는 순간 컴파일 에러가 발생합니다.

```
==========substr==========
str1   .substr(4, 3)   o W
strstr1.substr(4, 3)   o W

str1   .substr(4, 11)   o World
strstr1.substr(4, 11)   o World

==========find==========
str1   .find(str2)    #str2  =   "o":   4
strstr1.find(strstr2) #strstr2 = "o":   4

str1   .find(str2, 5)    #str2  =   "o":   7
strstr1.find(strstr2, 5) #strstr2 = "o":   7

str1   .find(str2)    #str2  =   "Zo":   18446744073709551615
strstr1.find(strstr2) #strstr2 = "Zo":   18446744073709551615

==========stoi==========
stoi(str1, &sz)    #str1  =   "\n\t+10bus" ->   num: 10   sz: 5
stoi(strstr1, &sz) #strstr1 = "\n\t+10bus" ->   num: 10   sz: 5

stoi(str1, &sz)    #str1  =   "\n\t -33bus" ->   num: -33   sz: 6
stoi(strstr1, &sz) #strstr1 = "\n\t -33bus" ->   num: -33   sz: 6

==========stof==========
stoi(str1, &sz)    #str1  =   "\n\t+10.1284abc" ->   num: -33   sz: 10
stoi(strstr1, &sz) #strstr1 = "\n\t+10.1284abc" ->   num: -33   sz: 10

stof(str1, &sz)    #str1  =   "\n\t -23.4433abc" ->   num: -23.4433   sz: 11
stof(strstr1, &sz) #strstr1 = "\n\t -23.4433abc" ->   num: -23.4433   sz: 11

==========to_string==========
to_string(1423)         : 1423
oopstd::to_string(1423) : 1423

to_string(51.123123123)         : 51.123123
oopstd::to_string(51.123123123) : 51.123123
```

8) substr: substr 역시 pose 가 문자열 범위 내에 존재한다면 잘 작동하지만 범위를 초과해버리는 순간 컴파일 에러가 발생하니다.

9) find: 찾으려는 문자열이 존재하면 그 문자열이 시작하는 인덱스를 반환하고 찾으려는 문자열이 없으면 size_t 의 최대값을 반환합니다.

10) stoi: atoi 와 동작방식이 똑같지만 매개변수로 string 을 받는다는 차이가 있습니다. string 에 저장된 정수 형식의 문자열을 정수로 변환합니다.

11) stof: atof 와 동작방식이 똑같지만 매개변수로 string 을 받는다는 차이가 있습니다. string 에 저장된 실수 형식의 문자열을 정수로 변환합니다.

12) to_string: 인자로 정수를 전달하면 정수 형식으로 문자열이 저장된 string 이 반환되고 인자로 실수를 전달하면 실수 형식으로 문자열이 저장된 string 이 반환된다.

❑ 고찰

replace 와 substr 메소드에서 out of range 를 호출해야 하는 상황이 존재했는데 replace 에서는 *this 를 반환했고, substr 에서는 빈 string 을 반환하도록 했습니다.

replace 메소드를 구현할 때 pose 와 len 의 크기에 따라 새로 할당할 메모리의 크기를 정하는데 어려움이 있었다. pos+len 이 원래 문자열의 길이를 초과하는지 아닌지에 따라 배열의 크기를 다르게 해줬더니 코드가 잘 실행됐다.

stoi, stof 같은 경우, string 헤더파일을 추가해서 실험해본 결과, (white space), (부호), 숫자가 순서대로 잘 나열돼 있지 않으면 오류가 난다는 것을 알게 되어, oopstd::stoi, stof 는 그런 상황에서 -1 을 반환하도록 코드를 작성했습니다.

Program 4

❑ 문제 설명

두 행렬 A, B 에 대한 덧셈, 뺄셈, 곱셈 연산과 한 행렬 A 에 대한 스칼라 덧셈, 스칼라 뺄셈, 스칼라 곱셈, 스칼라 나눗셈을 진행하는 함수를 만드는 문제입니다. 행렬들을 세팅하기 위한 클래스 메소드들도 작성하고 코드가 잘 실행이 되는지 실험하는 main 까지 작성하는 문제입니다.

add, sub 함수의 경우 Matrix a, b 의 크기가 다르다면 Matrix r 에 1x1 크기의 이차원 배열에 0 을 채워 넣고, "size of Matrixes are different."라는 문구를 출력합니다. mul 함수의 경우 a 의 cols 와 b 의 rows 의 크기가 같지 않으면 "The size of Matrixes should be (n x m), (m x l)"라는 문구를 출력하고 역시 Matrix r 에는 1x1 크기의 이차원 배열에 0 을 채워 넣었습니다. elementAdd, elementSub, elementMul, elementDiv 함수는 행렬의 크기와 상관 없이 항상 수행 가능하므로 실수 v 를 이차원 배열 각 요소에 연산하도록 작성했습니다. transpose 함수는 for 문으로 r 의 i, j 요소에 m 의 j, i 요소를 저장하도록 작성했습니다. deter 함수는 이차원배열 data 와 배열의 크기 size 를 인자로 받아 재귀를 수행하는데 이때, 재귀의 인자로 현재 참조중인 행과 열 인덱스를 제외한 나머지 요소들을 저장하고 있는 이차원 배열 subMat 와 크기 size-1 를 전달합니다. 재귀가 진행되다가 size 가 1 이 되면 그 값을 반환하도록 설계했습니다. adjoint 함수는 4 중 for 문으로 현재 참조중인 행과 열 인덱스를 제외한 나머지 요소들을 subMat 에 저장하고 deter(subMat, size-1)을 통해 구한 행렬식을 이용하여 여인수행렬을 구하고 transpose 함수를 통해 여인수행렬을 뒤집어 만들어진 수반행렬을 Matrix r 에 저장합니다. inverse 함수는 Div(r, 수반행렬, 행렬식) 을 수행하여 Matrix r 에 역행렬을 저장합니다.

❑ 결과 화면

A: 3 x 3, B: 3 x 3

```
Program Exit: 0, Continue: any input
1
rows and cols of A: 3 3
rows and cols of B: 3 3

A:
9.10    9.70    5.40
8.40    1.70    0.50
3.10    6.50    7.80

B:
8.00    1.40    3.40
5.50    1.00    8.00
3.40    1.00    4.20
```

```
Add of Matrixed:
17.10   11.10   8.80
13.90   2.70    8.50
6.50    7.50    12.00

Sub of Matrixed:
1.10    8.30    2.00
2.90    0.70    -7.50
-0.30   5.50    3.60

Mul of Matrixed:
144.51  27.84   131.22
78.25   13.96   44.26
87.07   18.64   95.30
```

```
Add scalar 5:
14.10   14.70   10.40
13.40   6.70    5.50
8.10    11.50   12.80

Sub scalar 5:
4.10    4.70    0.40
3.40    -3.30   -4.50
-1.90   1.50    2.80

Mul scalar 5:
45.50   48.50   27.00
42.00   8.50    2.50
15.50   32.50   39.00

Div scalar 5:
1.82    1.94    1.08
1.68    0.34    0.10
0.62    1.30    1.56
```

```
Transpose of A:
9.10    8.40    3.10
9.70    1.70    6.50
5.40    0.50    7.80

determinant of A:
-263.04

adjoint of A:
10.01   -40.56  -4.33
-63.97  54.24   40.81
49.33   -29.08  -66.01

Inverse of A:
-0.04   0.15    0.02
0.24    -0.21   -0.16
-0.19   0.11    0.25
```

1) 행렬 덧셈

$$\begin{pmatrix} 9.10 & 9.70 & 5.40 \\ 8.40 & 1.70 & 0.50 \\ 3.10 & 6.50 & 7.80 \end{pmatrix} + \begin{pmatrix} 8.00 & 1.40 & 3.40 \\ 5.50 & 1.00 & 8.00 \\ 3.40 & 1.00 & 4.20 \end{pmatrix} = \begin{pmatrix} 17.1 & 11.1 & 8.8 \\ 13.9 & 2.7 & 8.5 \\ 6.5 & 7.5 & 12 \end{pmatrix}$$

2) 행렬 뺄셈`

$$\begin{pmatrix} 9.10 & 9.70 & 5.40 \\ 8.40 & 1.70 & 0.50 \\ 3.10 & 6.50 & 7.80 \end{pmatrix} - \begin{pmatrix} 8.00 & 1.40 & 3.40 \\ 5.50 & 1.00 & 8.00 \\ 3.40 & 1.00 & 4.20 \end{pmatrix} = \begin{pmatrix} 1.1 & 8.3 & 2 \\ 2.9 & 0.7 & -7.5 \\ -0.3 & 5.5 & 3.6 \end{pmatrix}$$

3) 행렬 곱셈

$$\begin{pmatrix} 9.10 & 9.70 & 5.40 \\ 8.40 & 1.70 & 0.50 \\ 3.10 & 6.50 & 7.80 \end{pmatrix} \cdot \begin{pmatrix} 8.00 & 1.40 & 3.40 \\ 5.50 & 1.00 & 8.00 \\ 3.40 & 1.00 & 4.20 \end{pmatrix} = \begin{pmatrix} 144.51 & 27.84 & 131.22 \\ 78.25 & 13.96 & 44.26 \\ 87.07 & 18.64 & 95.3 \end{pmatrix}$$

4) 스칼라 곱셈

$$5 \cdot \begin{pmatrix} 9.10 & 9.70 & 5.40 \\ 8.40 & 1.70 & 0.50 \\ 3.10 & 6.50 & 7.80 \end{pmatrix} = \begin{pmatrix} 45.5 & 48.5 & 27 \\ 42 & 8.5 & 2.5 \\ 15.5 & 32.5 & 39 \end{pmatrix}$$

5) 전치행렬

$$\begin{pmatrix} 9.10 & 9.70 & 5.40 \\ 8.40 & 1.70 & 0.50 \\ 3.10 & 6.50 & 7.80 \end{pmatrix}^T = \begin{pmatrix} 9.1 & 8.4 & 3.1 \\ 9.7 & 1.7 & 6.5 \\ 5.4 & 0.5 & 7.8 \end{pmatrix}$$

6) 행렬식

$$\begin{vmatrix} 9.10 & 9.70 & 5.40 \\ 8.40 & 1.70 & 0.50 \\ 3.10 & 6.50 & 7.80 \end{vmatrix} = -263.04$$

7) 수반행렬, 역행렬

$$A^{(-1)} = \frac{1}{\det(A)} \cdot C^T = \frac{1}{-263.04} \cdot \begin{pmatrix} 10.01 & -40.56 & -4.33 \\ -63.97 & 54.24 & 40.81 \\ 49.33 & -29.08 & -66.01 \end{pmatrix} = \begin{pmatrix} -0.04 & 0.15 & 0.02 \\ 0.24 & -0.21 & -0.16 \\ -0.19 & 0.11 & 0.25 \end{pmatrix}$$

A, B 행렬이 정사각행렬일 때 모든 연산이 잘 수행되는 것을 확인할 수 있습니다.

A: 3 x 5, B: 5 x 4

```
Program Exit: 0, Continue: any input
1
rows and cols of A: 3 5
rows and cols of B: 5 4

A:
6.60    1.00    3.60    1.30    5.10
4.30    8.00    0.80    0.90    3.80
2.70    2.00    0.90    5.10    0.20

B:
6.40    2.50    7.00    5.40
1.00    5.90    1.30    6.30
8.30    4.50    9.30    4.70
9.00    2.70    6.80    4.50
2.80    0.90    7.80    7.20
```

```
Add of Matrixed:
size of Matrixes are different.
0.00

Sub of Matrixed:
size of Matrixes are different.
0.00

Mul of Matrixed:
99.10   46.70   129.60  101.43
60.90   67.40   83.70   108.79
73.21   36.55   66.11   55.80
```

```
Add scalar 5:
11.60   6.00    8.60    6.30    10.10
9.30    13.00   5.80    5.90    8.80
7.70    7.00    5.90    10.10   5.20

Sub scalar 5:
1.60    -4.00   -1.40   -3.70   0.10
-0.70   3.00    -4.20   -4.10   -1.20
-2.30   -3.00   -4.10   0.10    -4.80

Mul scalar 5:
33.00   5.00    18.00   6.50    25.50
21.50   40.00   4.00    4.50    19.00
13.50   10.00   4.50    25.50   1.00

Div scalar 5:
1.32    0.20    0.72    0.26    1.02
0.86    1.60    0.16    0.18    0.76
0.54    0.40    0.18    1.02    0.04
```

```
Transpose of A:
6.60    4.30    2.70
1.00    8.00    2.00
3.60    0.80    0.90
1.30    0.90    5.10
5.10    3.80    0.20

determinant of A:
It's not a square Matrix
0.00

adjoint of A:
It's not a square Matrix.
0.00

Inverse of A:
It's not a square Matrix.
0.00
```

1) 행렬 덧셈 뺄셈

A, B 의 크기가 같지 않아 연산이 진행되지 않습니다.

2) 행렬 곱셈

$$\begin{pmatrix} 6.60 & 1.00 & 3.60 & 1.30 & 5.10 \\ 4.30 & 8.00 & 0.80 & 0.90 & 3.80 \\ 2.70 & 2.00 & 0.90 & 5.10 & 0.20 \end{pmatrix} \cdot \begin{pmatrix} 6.40 & 2.50 & 7.00 & 5.40 \\ 1.00 & 5.90 & 1.30 & 6.30 \\ 8.30 & 4.50 & 9.30 & 4.70 \\ 9.00 & 2.70 & 6.80 & 4.50 \\ 2.80 & 0.90 & 7.80 & 7.20 \end{pmatrix} = \begin{pmatrix} 99.1 & 46.7 & 129.6 & 101.43 \\ 60.9 & 67.4 & 83.7 & 108.79 \\ 73.21 & 36.55 & 66.11 & 55.8 \end{pmatrix}$$

3) `스칼라 곱셈

$$5\cdot\begin{pmatrix} 6.60 & 1.00 & 3.60 & 1.30 & 5.10 \\ 4.30 & 8.00 & 0.80 & 0.90 & 3.80 \\ 2.70 & 2.00 & 0.90 & 5.10 & 0.20 \end{pmatrix}=\begin{pmatrix} 33 & 5 & 18 & 6.5 & 25.5 \\ 21.5 & 40 & 4 & 4.5 & 19 \\ 13.5 & 10 & 4.5 & 25.5 & 1 \end{pmatrix}$$

4) 스칼라 나눗셈

$$\frac{1}{5}\cdot\begin{pmatrix} 6.60 & 1.00 & 3.60 & 1.30 & 5.10 \\ 4.30 & 8.00 & 0.80 & 0.90 & 3.80 \\ 2.70 & 2.00 & 0.90 & 5.10 & 0.20 \end{pmatrix}=\begin{pmatrix} 1.32 & 0.2 & 0.72 & 0.26 & 1.02 \\ 0.86 & 1.6 & 0.16 & 0.18 & 0.76 \\ 0.54 & 0.4 & 0.18 & 1.02 & 0.04 \end{pmatrix}$$

5) 전치행렬

$$\begin{pmatrix} 6.60 & 1.00 & 3.60 & 1.30 & 5.10 \\ 4.30 & 8.00 & 0.80 & 0.90 & 3.80 \\ 2.70 & 2.00 & 0.90 & 5.10 & 0.20 \end{pmatrix}^{T}=\begin{pmatrix} 6.6 & 4.3 & 2.7 \\ 1 & 8 & 2 \\ 3.6 & 0.8 & 0.9 \\ 1.3 & 0.9 & 5.1 \\ 5.1 & 3.8 & 0.2 \end{pmatrix}$$

6) 행렬식, 수반행렬, 역행렬

A 가 정사각행렬이 아니어서 연산이 불가능합니다.

마지막으로 A 의 cols 와 B 의 rows 가 같지 않아 곱연산이 불가능한 경우를 보이겠습니다.

```
rows and cols of A: 3 5
rows and cols of B: 4 4

A:
2.90    5.70    7.70    8.00    9.20
2.00    0.50    0.20    4.60    8.40
7.10    1.60    0.10    0.60    0.80

B:
2.50    2.30    4.10    4.60
0.50    8.40    5.50    2.30
2.50    2.50    7.40    1.10
1.90    2.90    9.70    4.10
```

```
Mul of Matrixed:
The size of Matrixes should be (n x m), (m x l)
0.00
```

❑ 고찰

행렬식을 구하는 과정에서 재귀함수를 사용했습니다. 처음 매개변수로 전달한 이차원 배열을 계속해서 재귀함수의 매개변수로 전달하여 문제를 해결하려고 시도했으나 현재 참조하고 있는 행, 열과 같은 행, 열 인덱스를 가지고 있는 요소들을 제외한 나머지로 재귀를 해야 했습니다. 따라서 재귀함수 내부에서 크기를 1 줄인 이차원 배열을 새로 할당한 뒤 적절한 값으로 초기화하여 재귀함수의 매개변수로 전달하도록 해서 문제를 해결했습니다.