다음에 열거된 요구사항(requirement)를 충분히 숙지하고 과제를 제출하시오.

1. 행렬의 행(row)과 열(column)의 값을 입력 받는다. (`scanf()` 사용)
2. 입력받은 행(row)과 열(column)의 값을 함수구현에 활용한다.
3. 모든 행렬은 동적 메모리할당(dynamic memory allocation) 방식으로 생성한다. (함수: `create_matrix()`)
4. 첨부된 `matrix.c` 코드에 정의된것과 같이 메뉴 시스템을 사용하여 구현한다.

```
[----- [Your Name]  [Student ID] -----]
Input row and col : 3 3
Matrix Created.
------------------------------------------------------------------
                     Matrix Manipulation
------------------------------------------------------------------
Initialize Matrix   = z           Print Matrix        = p
Add Matrix          = a           Subtract Matrix     = s
Transpose matrix_a  = t           Multiply Matrix     = m
Quit                = q
------------------------------------------------------------------
Command =
```

5. Initialize Matrix - 행렬 A, B의 성분값(entry value)은 $0 \sim 19$ 사이의 값을 갖는 랜덤값(random value)으로 채운다. (`rand()` 사용) (함수: `fill_data()`)
6. Print Matrix - A와 B 행렬을 출력한다. (`print_matrix()` 사용)
7. Add Matrix - $A + B$를 구현한다. (함수: `addition_matrix()`)
8. Subtract Matrix - $A - B$를 구현한다. (함수: `subtraction_matrix()`)
9. Transpose matrix_a - A의 전치행렬 T를 구현한다. (함수: `transpose_matrix()`)
10. Multiply Matrix - $A \times T$를 구현한다. (함수: `multiply_matrix()`)
11. Quit를 제외한 모든 메뉴는 다시 실행될 수 있어야 한다.
12. 모든 함수는 전처리 및 후처리 검사(pre/post-condition test)를 수행하여 비정상적인 입력과 결과값에 대비 한다.

```
int** create_matrix(int row, int col) {

    /* check pre conditions */
    if(row <= 0 || col <=0) {
        printf("Check the sizes of row and col!\n");
        return NULL;
```

```
        }
        ....

        /* check post conditions */
        if(matrix == NULL) {
            /* proper actions for unexpected conditions */
        }

        return matrix;
    }
```

13. 모든 메뉴에서 연산이 끝나면, `print_matrix()`를 사용하여 결과값을 프린트한다.
14. 연산이 종료되거나 프로그램을 종료할 때 할당했던 메모리를 해제 한다. (함수: `free_matrix()`)
15. `return`이 정의된 경우 정상종료와 비정상종료를 나타내는 값을 돌려준다.

```
....
if (everything is O.K) return 1;
return -1; /* Not O.K */
....
```

16. 아래 정의된 함수를 구현하여 프로그램을 완성한다. **정의된 함수는 수정할 수 없다.** 필요할 경우 추가적인 함수를 정의하고 구현은 가능.

```
/* create a 2d array whose size is row x col using malloc()
   and return the 2d array */
int** create_matrix(int row, int col);

/* free memory allocated by create_matrix() */
int free_matrix(int** matrix, int row, int col);

/* print matrix whose size is row x col */
void print_matrix(int** matrix, int row, int col);

/* assign random values to the given matrix */
int fill_data(int** matrix, int row, int col);

/* transpose the matrix to matrix_t */
int transpose_matrix(int** matrix, int** matrix_t, int row, int col);

/* matrix_sum = matrix_a + matrix_b */
int addition_matrix(int** matrix_a, int** matrix_b, int row, int col);

/* matrix_sub = matrix_a - matrix_b */
```

```
int subtraction_matrix(int** matrix_a, int** matrix_b, int row, int col);

/* matrix_axt = matrix_a x matrix_t */
int multiply_matrix(int** matrix_a, int** matrix_t, int row, int col);
```

17. GNU C Compiler + Open Source Editor를 사용한다.

18. **각 프로그램 소스에 주석을 남긴다.**

19. **소스파일에 성명, 학번이 실행시 출력되도록 한다**(`printf()`).

    ```
    [----- [Your Name]  [Student ID] -----]
    ```

20. 본인이 작성한 모든 프로그램을 GitHub에 업로드한다.

21. 작성된 소스파일을 보고서로 만든다.

22. 보고서에는 실행결과를 Screen Capture하여 첨부한다.

23. 프로그램 보고서에 본인의 GitHub URL을 명시한다.

24. 과제 결과물을 eCampus에 업로드한다.

25. **주의사항**

    (a) 마감시간을 넘기 모든 과제는 0점 처리됨.

    (b) 주석이 부실한 코드는 감점 대상이 되며, 주석 복사의 경우 모두 0점 처리됨.

    (c) GitHub를 이용하지 않은 과제는 0점 처리됨.

    (d) 컴파일 및 실행이 안될 경우 0점 처리됨.

    (e) 정상동작하지 않은 프로그램 감점 처리됨.