자료구조 2주차 과제

제출일: 2022-09-14
학번/이름: 2016121150 / 윤준영

문1) 다음은 n*n 2차원 배열의 대각선 원소의 합을 구하는 함수를 작성하여, 다음 프로그램이 15와 18을 출력하도록 완성하시오. 작성한 함수가 어떻게 임의의 크기의 n*n 함수에 대하여 동작하는지 자세히 설명하시오.

```
main() {
   int x[3][3]={{1,2,3},{4,5,6},{7,8,9}}
   int y[4][4]={{1,2,3,4},{2,3,4,5},{4,5,6,7},{5,6,7,8}}

   printf("%d\n", sumDiag(&(x[0][0]), 3));  //x[0]와 동일
   printf("%d\n", sumDiag(&(y[0][0]), 4));
}

int sumDiag(int *,           ) {



}
```

```
HW2 > C hw2_1.c > ...
1   #include <stdio.h>
2
3   // 1 : n*n 2차원 배열의 대각선 원소의 합을 구하는 함수를 작성하라.
4   //    ans : 15, 18
5
6   main() {
7       int x[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
8       int y[4][4] = {{1,2,3,4},{2,3,4,5},{4,5,6,7},{5,6,7,8}};
9
10      printf("%d\n", sumDiag(&(x[0][0]), 3)); //x[0]와 동일
11      printf("%d\n", sumDiag(&(y[0][0]), 4));
12  }
13
14  int sumDiag(int *a, int n){
15      int s = 0;
16      int i;
17      for (i = 0; i < n; i++)
18          s = s + *(a + (i * n) + i);
19      return s;
20  }
```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
15
18
* Terminal will be reused by tasks, press any key to close it.
```

함수 sumDiag는 n*n 배열의 첫 원소의 주소와 배열의 크기 n을 받게 된다.

대각선 원소의 합은 다음과 같이 표현할 수 있다.

$$sum = a[0][0] + a[1][1] + a[2][2] + \cdots + a[n-1][n-1]$$

여기서 a[i][i]의 주소는 다음과 같이 구할 수 있다.

$$\&(a[i][i]) = \&(a[0][0]) + (i \times n) + i$$

따라서 반복문을 이용하여 왼쪽과 같은 코드를 작성할 수 있다.

문2) 다음은 두 행렬의 곱을 계산하는 함수이다.
main함수를 추가하여 3x4 행렬 {{1,2,3,4},{2,3,4,5},{4,5,6,7}}과 4x2 행렬 {{1,2},{3,4},{5,6},{7,8}}의 곱을 계산하여 다음과 같이 출력하는 프로그램을 작성하시오.

```
void multiply_matrix(int m1[], int m2[],
         int m3[], int m, int n, int l)
{
    int i, j, k;

    for (i = 0; i < m; i++)
        for (j = 0; j < l; j++) {
            m3[i*l+j] = 0;
            for (k = 0; k < n; k++)
                m3[i*l+j] += m1[i*n+k]*m2[k*l+j];
        }
}
```

(출력)
50 60
66 80
98 120

HW2 > C hw2_2.c > ...

```c
#include <stdio.h>

// 2 : main 함수를 추가하여 3x4 행렬 {{1,2,3,4},{2,3,4,5},{4,5,6,7}}과
//    4x2 행렬 {{1,2},{3,4},{5,6},{7,8}}의 곱을 계산하여 출력

void multiply_matrix(int m1[], int m2[], int m3[], int m, int n, int l)
{
    int i, j, k;

    for (i = 0; i < m; i++)
        for (j = 0; j < l; j++) {
            m3[i*l+j] = 0;
            for (k = 0; k < n; k++)
                m3[i*l+j] += m1[i*n+k]*m2[k*l+j];
        }
}

main() {
    int a[3][4] = {{1,2,3,4},{2,3,4,5},{4,5,6,7}};
    int b[4][2] = {{1,2},{3,4},{5,6},{7,8}};
    int c[3][2];
    int m=3, n=4, l=2;
    int i, j;
    multiply_matrix(&(a[0][0]),&(b[0][0]),&(c[0][0]),m,n,l);
    for (i=0; i<m; i++){
        for (j=0; j<l; j++)
            printf("%d ", c[i][j]);
        printf("\n");
    }
}
```

i = 0 ~ m-1
j = 0 ~ l-1
C의 i행 j열출력 — 행이 바뀔때 줄바꿈

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
50 60
66 80
98 120
* Terminal will be reused by tasks, press any key to close it.
```

문3) 다음 함수에 대하여 물음에 답하시오.

```
void transpose(int a[][3], int b[][3]) {
  int n=a[0][1]; int terms = a[0][2];
  int *s = (int *)calloc(n,sizeof(int));
  int *t = (int *)calloc(n,sizeof(int));
  int i,j;
  b[0][0]=n; b[0][1]=a[0][0]; b[0][2]=terms;
  for (i=1; i<=terms; i++) s[a[i][1]]++;
  t[0] = 1;
  for (i=1; i<n; i++) t[i]=t[i-1]+s[i-1];
  for (i=1; i<=terms; i++) {
     j = t[a[i][1]];
     b[j][0] = a[i][1];
     b[j][1] = a[i][0];
     b[j][2] = a[i][2];
     t[a[i][1]] = j+1;
  }
}
```

1) main함수를 작성하여 (0,0), (0,2), (1,3), (2,2), (3,0), (3,1)위치에 각각 1, 2, 3, 4, 5, 6이 저장된 2차원 4x4 행렬을 희소행렬 표현으로 저장하고, 위 transpose 함수를 적용한 전치 행렬을 출력하는 프로그램을 작성하시오. 문제 2)에서 사용한 행렬을 축적하는 코드를 이용하여 행렬을 출력하는 matprint 프로그램을 작성하였다.

```
HW2 > C hw2_3.c > ...
  1  #include <stdio.h>
  2  #include <stdlib.h>

 26  void matprint(int a[], int m, int l) {
 27      for (int i=0; i<m; i++){
 28          for (int j=0; j<l; j++)
 29              printf("%d ", a[i*l+j]);
 30          printf("\n");
 31      }
 32      printf("\n");
 33  }
 34
 35  main() {
 36      // 희소행렬의 크기 = (요소+1)X(차원+1) >> 7X3
 37      int a[7][3] = {{4,4,6},{0,0,1},{0,2,2},{1,3,3},{2,2,4},{3,0,5},{3,1,6}};
 38      int b[7][3];
 39      transpose(a, b);
 40      matprint(&(b[0][0]), 7, 3);
 41  }
```

원본행렬 a

4	4	6
0	0	1
0	2	2
1	3	3
2	2	4
3	0	5
3	1	6

→

전치행렬 b

4	4	6
0	0	1
0	3	5
1	3	6
2	0	2
2	2	4
3	1	3

```
4 4 6
0 0 1
0 3 5
1 3 6
2 0 2
2 2 4
3 1 3

*  Terminal will be reused by tasks, press any key to close it.
```

2) 위 1)의 입력에 대한 위 함수 수행 시 생성되는 s와 t행렬을 쓰고, 두 행렬의 값들의 의미를 설명하시오. (예: s[2]가 3인 것은 입력 행렬에 0이 아닌 것이 3개이기 때문이다...(<-정답은 아님))

S=[2,1,2,1]

t=[1,3,4,6] (t 생성 시점) t=[3,4,6,7] (함수종료시점)

S행렬은 희소행렬의 2번째 열이 이루어진 요소의 개수를 나타낸다.
S[i]의 요소는 희소행렬의 2번째열에 있는 i 값의 개수를 나타낸다.

t행렬은 t[i]에서 희소행렬의 2번째열의 값이, i값 인덱 전치행렬에 작성되는 행의 위치를 나타낸다.

3) 위 함수를 호출하면 수행 후에 heap 메모리에 가비지(garbage)가 남는다. 그 이유를 설명하고, 가비지가 생성되지 않도록 프로그램을 수정하시오.

calloc 함수를 사용하면 heap 영역에 메모리에 특정 크기의 공간을 할당하고 0으로 초기화한다.
heap 영역의 메모리는 함수가 종료되어도 사라지지 않으므로 사용후 할당을 해제하지 않으면 가비지가 남게 된다.
따라서 free( )를 이용하여 메모리 할당을 해제하여야 한다.

다른 방법은 calloc을 사용하지 않고 반복문을 사용하여 초기화하는 방법도 있다.

```
void transpose(int a[][3], int b[][3]) {
    int n=a[0][1]; int terms = a[0][2];
    int *s = (int *)calloc(n,sizeof(int));
    int *t = (int *)calloc(n,sizeof(int));
    int i,j;
    b[0][0]=n; b[0][1]=a[0][0]; b[0][2]=terms;
    for (i=1; i<=terms; i++) s[a[i][1]]++;
    t[0] = 1;
    for (i=1; i<n; i++) t[i]=t[i-1]+s[i-1];
    for (i=1; i<=terms; i++) {
        j = t[a[i][1]];
        b[j][0] = a[i][1];
        b[j][1] = a[i][0];
        b[j][2] = a[i][2];
        t[a[i][1]] = j+1;
    }
    free(s);
    free(t);
}
```

할당해제

```
void transpose2(int a[][3], int b[][3]) {
    int n=a[0][1]; int terms = a[0][2];
    int s[n], t[n];
    int i,j;
    for (i=0; i<n; i++) {
        s[i] = 0;
        t[i] = 0;
    }
    b[0][0]=n; b[0][1]=a[0][0]; b[0][2]=terms;
    for (i=1; i<=terms; i++) s[a[i][1]]++;
    t[0] = 1;
    for (i=1; i<n; i++) t[i]=t[i-1]+s[i-1];
    for (i=1; i<=terms; i++) {
        j = t[a[i][1]];
        b[j][0] = a[i][1];
        b[j][1] = a[i][0];
        b[j][2] = a[i][2];
        t[a[i][1]] = j+1;
    }
}
```

반복문을 사용하여 초기화