1 번:

Sparse Matrix 는 많은 항들이 0 으로 구성되어 있는 행렬이다. 구현이 간단하지만 메모리의 낭비가 심하다는 단점이 있다. Sparse Matrix 에서 0 이 0 아닌 노드만을 표현한 방법이 dense Matrix 이다. Dense Matrix 는 프로그래밍이 복잡하지만 메모리를 아낄 수 있다.

1) 해결해야 하는 문제:

For a sparse matrix represented by storing only non-zero elements, implement a transpose operation.

e.g. A: $m \times n$ matrix → A^T: $n \times m$ matrix

2) 해결 방법:

거의 0 의 값을 가지고 있는 sparse matrix 를 0 이 아닌 노드만을 저장하는 dense matrix 로 표현한다. Dense matrix 는 row,column,value 를 하나의 행으로 구성된다. 이 matrix 를 transpose matrix 로 만들려면 dense matrix 에서 row 값과 column 값을 서로 바꾸고 value 값은 그대로 가져온다. 이때 구조체를 사용하여 row, col, value 값을 묶어서 쓸 수 있게 하고 바뀐 transpose matrix 가 낮은 행에서 높은 행 순서대로 출력되야 한다. 열이 행으로 바뀌기 때문에 구조체 bt 를 선언해서 그대로 대입하면 낮은 열에서 높은 열 순서대로 출력되기 때문에 구조체 b 의 col 값이 0 열인 요소를 찾아서 구조체 bt 의 row 값의 0 행으로 저장하는 식(1 열,2 열,3 열...->1 행, 2 행, 3 행)으로 진행해야 한다.

3) 구현:

For 문을 써서 b 의 row 값과 증가하는 i 의 값이 일치하면 bt matrix 에 저장되는 식으로 코드를 작성하였다. Sparse Matrix 구조체는 element 구조체를 포함하고 있는데 element 는 row, col, value 값을 저장하고 있어서 b.elemet.value 식으로 값을 가져올 수 있다. 완성된 dense matrix 를 다시 sparse matrix 로 바꾸기 위해서 matrix_printf_s 라는 함수를 만들었는데 col 이 증가 후 도 중첩하여 증가하는 식으로 진행하고 dense matrix 에서 row, col 값이 증가하는 i,f 값과 일치하면 value 값을 출력하고 아니면 0 을 출력하는 방식으로 작성하였다. 이때 0 이 아닌 value 값을 출력하면 0 으로 초기화한

k 값이 증가하게 되고 다음 k 번째 row, col 값과 일치하는 I,j 를 찾게 된다. 행의 순서 즉 저장 순서대로 출력할 수 있다.

2 번

Dynamic Memory Allocation 은 메모리를 할당하는 방법 중 하나이다. Static Memory Allocation 은 메모리 사이즈가 고정되어 있어서 고정된 값보다 더 큰 값이 들어오면 처러하지 못하고 더 작은 입력이 들어온다면 메모리가 낭비되는 단점을 가지고 있다 그에 비해 Dynamic Memory Allocation 은 메모리를 효과적으로 사용할 수 있다. 메모리를 운영체제로부터 할당받아서 사용하고 사용이 끝나면 메모리를 반환한다.

1) 해결해야 하는 문제:

Implement a function 'mem_alloc_3D_double' of allocating 3D array of double

2} 해결 방법:

Double 형 3 중 포인터 A,B,C 를 선언하고 각각 동적 메모리를 할당해준다. 동적 메모리를 사용하여 A,B 의 행,열,높이의 위치가 같은 것 끼리 서로 더해준다. 항들을 순서대로 출력해준 후 동적 메모리를 시스템에 반납한다.

3) 구현

double ***A = (double***)malloc(X * sizeof(double **));식으로 1st column 을 만든다. 이때 sizeof(double)*X 사이즈가 할당된다. X 사이즈가 바뀌어도 시스템은 돌아갈 수 있게 된다 for 문을 돌려서 column 을 row 개 생산하여 준다 이때 마찬가지로 sizeof(double)*Y 사이즈가 할당된다 3d 이므로 같은 방법으로 sizeof(double)*z 사이즈가 할당된 높이도 쌓아주면 된다. 만든 배열들을 3 중 for 문을 돌려서 각각 랜덤 값을 입력하고 같은 위치끼리 더하고 출력하면 문제를 해결할 수 있다.