`c[x][y] = a[x][y] + b[x][y]` ∴ Matrix c correctly stores the sum of matrices a and b. atleast for its 1st rows * cols elements.

(2) Rewrite the matrix add function using dynamically allocated arrays. The header for the function should be: void add(int **a, int **b, int **c, int rows, int cols)

```
void add(int **a, int **b, int **c, int rows, int cols)
{ int i,j;
  MALLOC(c, rows*sizeof(*c));
  for(i=0; i<rows; i++)
    CALLOC(c[i], cols*sizeof(**c));
  for(i=0; i<rows; i++)
    for(j=0; j<cols; j++)
      c[i][j] = a[i][j] + b[i][j];
}
```

4) 1. void mult(int a[][MAX_SIZE], int b[][MAX_SIZE], int c[][MAX_SIZE])

```
2. {
3.   int i,j,K;
4.   for(i=0; i<MAX_SIZE; i++)
5.     for(j=0; j<MAX_SIZE; j++) {
6.       c[i][j]=0;
7.       for(K=0; K<MAX_SIZE; K++)
8.         c[i][j] += a[i][K] * b[K][j];
9.     }
10. }
```

Loop(K) Invariant

Just before the start of the iteration when K=l, the element of matrix c at row i, column j stores the sum of the respective products of the xth element of row i of matrix a and xth element of column j of matrix b for all x, $0 \le x < l$, $x \in N$

Initialization: K=0. On line 7, if K=0, it indicates the state of being just before the 1st iteration of the loop. ∵ l=0, the range $0 \le x < 0$ doesn't make sense. On line 6, we have already initialized 0 to c[i][j]. ∴ Loop invariant trivially holds.

Maintenance: Let, just before the iteration when K=l, the loop invariant holds.