Exercise 3.1 — ASCII Art — Ribbons (60% of Midterm 2, 2003)
Throughout this question, the second dimension of the two-dimensional arrays will be some fixed WIDTH; in the examples this is 30.

(a) Implement a C function printCharArray that prints the contents of a two-dimensional character array to the screen, each row on a separate line.

(b) Implement a C function putRibbon that, given a two-dimensional character array, a start height \( h \) and a character \( c \), will place a “ribbon” of \( c \) values into the array that starts at height \( h \) and then wind upwards diagonally around the array.

Below, the first box contains the result of putting a ribbon of asterisks from start height 2 into a \( 7 \times 30 \) array filled with space characters; the second box contains the result of additionally inserting a ribbon of plus characters.

(c) Implement a C function putSlantedRibbon that in addition to the arguments of putRibbon also accepts an integral slant value that indicates the steepness of the ribbon’s slant as it winds around the array. This allows one to produce the contents of the box to the right with a single call to putSlantedRibbon with a \( 7 \times 30 \) array filled with space characters.

As an additional feature, this function must not override non-space characters in the array. Use an auxiliary function squeeze to squeeze a character into its target position, pushing right all non-space content encountered at the target position and at consecutive positions — the first space character encountered will be consumed.

The same call as for the previous box, when applied after the second box of (b), produces the box to the right — observe how the “A” characters sometimes push only a “+” to the right, sometimes the combination “*+”; at the end of the third line, a “+” has been “pushed off the board”.

(d) Write a main program that uses the above (and other) functions to produce as screen output the contents of the four example boxes above in the same sequence as above, using a single array of size \( 7 \times 30 \).

Do not forget design and documentation, in particular interface documentation for functions!
Exercise 3.2 — Simulation of C Program Execution  (40% of Midterm 2, 2003)

Simulate execution of the following correct ANSI C program; show the intermediate steps and show which output is produced:

```c
#include <stdio.h>
define SIZE 5
int q[SIZE];

void printq() {
    printf("[ ");
    int i;
    for (i=0; i < SIZE; i++)
        printf("%4d ", q[i]);
    printf("\n");
}

int s (int i, int j) {
    int k = q[i];
    q[i] = j;
    return k;
}

int f (int m, int n) {
    int h, r, mm = m + 1, nn = n - 1;
    printf("f(%d,%d) --- ", m, n);
    printq();
    if (m >= n) return q[m];
    h = s(mm, q[m]);
    r = f(mm, nn);
    q[m] = s(nn, q[n]);
    q[n] = h;
    return r;
}

int main() {
    int i;
    printf("How often?\n");
    scanf("%d", &count);
    for (i=1; i <= count; i++)
        printf("Hello!\n");
    return 0;
}
```

Exercise 3.3 — Compilation Phases  (8% of Midterm 1, 2004)

Name the phases of compilation — give a short description, too — and the result of each phase.

Exercise 3.4 — Find Errors  (16% of Midterm 1, 2004)

In each of the following programs or program segments,

• Find and describe the error. If the error can be corrected, explain how.

• Mark any unclear or unintuitive use of C features, explain the problem, and propose improvements.

(a) int p=1, q=2.3;  
p = q = 7;
printf ("q = %s\n", q );

(b) int funny ( int n, int k ) {
    return n ? k * funny (n-1) : 1;
}

(c) int strange (int q, int r) {
    int m;
    if ( q < r )
        m = r;  /* set m */
    else    /* to minimum */
        m = q;  /* of q and r */
    return m * m + q * r;
}

(d) #include <stdio.h>
int main() {
    int i, count;
    printf("How often?\n");
    scanf("%d", &count);
    for( i=1; i <= count; i++)
        printf("Hello!\n");
    return 0;
}