Exercise 2.1 — Treasure Hunt (45% of Midterm 1, 2003)

**Design** and implement a C program to play the “blind” board game “treasure hunt”.

- The board has $20 \times 20$ fields, from $(1, 1)$ to $(20, 20)$.
- On field $(17, 2)$ there is a treasure.
- The player starts on field $(9, 10)$, but is not told this.
- All fields $(x, y)$ with $(x + 2y)$ divisible by $5$ are **forbidden**, i.e., the player must not be allowed to move onto such a field.
- The player navigates the board by entering “numeric keypad cursor control commands”:
  - “2” moves **down** one step
  - “8” moves **up** one step
  - “4” moves **left** one step
  - “6” moves **right** one step

  After each successful move, **only** the new distance to the treasure is displayed — for this, the 1-norm is used and whether a field is forbidden or not does not matter, so, e.g., the distance from $(9, 10)$ to $(17, 3)$ is $15$ (calculated as $8 + 7$).

- When the player tries to move off the board or onto a forbidden field, a message is displayed noting that the move is impossible, but **not** why it is impossible.
- When the player moves to the field where the treasure is, a congratulatory message is displayed and the program terminates.

**Assume that the user will input only numbers! Do not use arrays!**

Exercise 2.2 (Textbook Exercise Recommendation)

Read chapter 4 of the textbook. Do **at least** the following exercises: 4.5–4.14, 4.24, 4.29
Exercise 2.3 — ASCII Art: Zig-Zag — (50% of Midterm 1, 2004)

Design and implement a C program that asks the user for a height, and for two offset numbers, and uses these three numbers to print a combination of two zig-zag lines of the same height, as in the following example:

```
+         X       +         X       +         X       +         X       +
+ +       X X     + +       X X     + +       X X     + +       X X     + +
+   +     X   X   +   +     X   X   +   +     X   X   +   +     X   X   +   +
+     +   X     X +     +   X     X +     +   X     X +     +   X     X +
*       + X       *       + X       *       + X       *       + X       *
X       *       + X       *       + X       *       + X       *       + X
X     X +     +   X     X +     +   X     X +     +   X     X +     +   X
X   X   +   +     X   X   +   +     X   X   +   +     X   X   +   +     X
X X     + +       X X     + +       X X     + +       X X     + +       X X
```

Note that one of the zig-zag lines is drawn using the “plus” symbol, the other using the letter “X”, and where both zig-zag lines intersect, the asterisk “*” is used.

The grid lines are of course not part of the output. Here is another example without those grid lines — any such pattern should be producable:

```
  +   X               +   X               +   X               +   X
  + + X X             + + X X             + + X X             + + X X
  +   *   X           +   *   X           +   *   X           +   *   X
   X +   X         +   X +   X         +   X +   X         +   X +   X
  X   +   X       +   X   +   X       +   X   +   X       +   X   +   X       +
  X     +   X     +   X     +   X     +   X     +   X     +   X     +   X     +
   X       +   X   +   X       +   X   +   X       +   X   +   X       +   X   +
  +   X +   X         +   X +   X         +   X +   X         +   X +
  +   *   X           +   *   X           +   *   X           +   *   X
  + + X X             + + X X             + + X X             + + X X
  +   X               +   X               +   X               +   X
```

Assume that the user will input only numbers! Do not use arrays!

Decompose into functions! Design and Document!

Exercise 2.4

What is the output of the following C program (which prints not more than ten lines):

```
#include <stdio.h>
int main ( void ) {
  char input[] = "terasse";
  char result[] = "   "; // six spaces
  int i, j = 0, c = 3, q;
  for ( q = 3; q ≥ 0; q = q - c ) {
    for ( i = 0; i < c; i++ ) {
      printf("j = %d	c = %d	q = %d	i = %d\n", j, c, q, i);
      result[j] = input[q + i];
      j = j + 1;
    }
    c = c - 1;
  }
  printf("%s\n", result);
  return 0;
}
```

What is the value of `q` after termination of the outer loop?