1. Intro. The description in (the Romanian translation of) TAoCP of the algorithms for computing the inverse of a permutation in-place is ugly and hard to understand. I try to describe here the method that I came up with after reading the problem. I do not know yet if this is the same as **Algorithm I** in the book.

We will travel cycles in increasing order of their leaders, and the leaders are always the smallest element of the cycle. Whenever we see we see $p[x] = y$ we set $p[y] = x$, but after remembering the old value of $p[y]$. Once we set a value in p we mark it. These markings are used to detect the end of cycles.

```
#include <stdio.h>
  int p[100];   /* the permutation */
  int m[100];   /* markings */
  int n;   /* size of the permutation */

  int main()
  {
    int x, y, py;

    scanf("%d␣", &n);
    for (x = 0; x < n; ++x) scanf("%d␣", &p[x]);
    x = 0;
    while (1) {
      while (x < n ∧ m[x]) ++x;
      if (x ≡ n) break;
      y = p[x], py = p[y];
      while (¬m[y]) {
        p[y] = x, m[y] = 1;
        x = y, y = py, py = p[y];
      }
    }
    for (x = 0; x < n; ++x) printf("%d␣", p[x]);
    printf("\n");
  }
```

2. Note that only one bit of m is used so the program can be optimized (as it is in TAoCP).

3. Index.

INVPERM