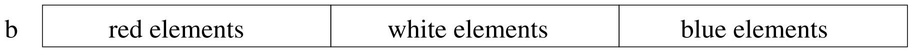
CS310: Paradigms of Programming
Homework 2

Solve each of the following problems using iteration by first writing a post-condition and then developing a loop invariant from it. You may assume that simple procedures like 'swap' are already available.

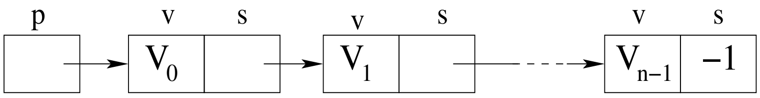
1. Write an iterative version of Mergesort by first writing the post-condition and then developing a loop invariant from it. Assume that the "Merge" routine is already implemented and that you are sorting elements stored in an array (as opposed to a linked list done in class).

2. (The Dutch National Flag). Given is an array $b[0:n-1]$ for fixed $n \geq 0$, each element of which is colored either red, white or blue. Write a program to permute the elements so that all the red elements are first and all the blue ones last. That is, the program is to establish



The color of an element may be tested with Boolean expressions $red(b[i])$, $white(b[i])$ and $blue(b[i])$, which return the obvious values. The number of such tests should be kept to a minimum. The only way to permute array elements is to swap two of them; the program should make at most n swaps.

3. (Link Reversal). A simple variable p and two arrays $v[0:?]$ and $s[0:?]$ are used to contain a sequence of values $V_0, V_1, ..., V_{n-1}$ as a *linked list*:



That is,

(1) $v[p]$ contains the first value V_0;

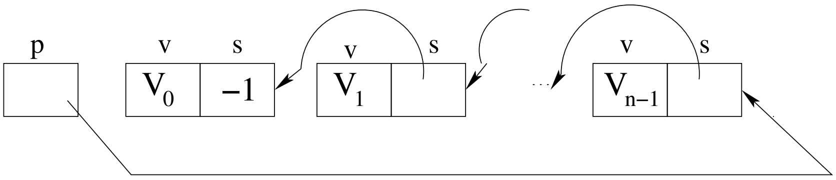
(2) for $0 \leq i < n-1$, if $v[k]$ contains the value V_i, then $v[s[k]]$ contains the value V_{i+1};

(3) if $v[k]$ contains the last value V_{n-1}, then $s[k] = -1$.

No ordering of values in array elements is implied. For example, the fact that V_0 is followed by V_1 in the linked list does not mean that $v[p+1]$ contains V_1. Write a program that reverses the links -the arrows implemented by array s. Array v should not be altered, and upon termination the linked list should be:

4. *Writing a Value as the Sum of Squares.* Write a program that, given a fixed integer $r \geq 0$, generates all different ways in which r can be written as the sum of two squares -i.e. that generates all pairs (x, y) satisfying

$$x^2 + y^2 = r \wedge 0 \leq y \leq x \qquad (1)$$



To help in writing it (and to arrange to use the strategy of taking a relation out of a loop), assume the following. Two arrays xv and yv will hold the values of the pairs (x, y) satisfying (1). Furthermore, the pairs are to be generated in increasing order of their x-values, and a variable x is used to indicate that all pairs with x-value less than x have been generated. Thus, the first approximation to the invariant of the main loop of the program will be

$P1: 0 \leq i \wedge ordered(xv[0:i-1]) \wedge$
the pairs $(xv[j], yv[j]), 0 \leq j < i$, are all the pairs
with x-value $< x$ that satisfy (1).

5. Given is an array $g[0:N-1]$, $N \geq 2$, satisfying $0 \leq g[0] \leq ... < g[N-1]$. Define

$h_1 = g[0] + g[1]$
$h_k = h_{k-1} + g[k]$ for $1 < k \leq N-1$

Write a program to construct an array $X[0:2*N-1]$ containing the values

$g[0], ..., g[N-1], h_1, ..., h_{N-1}$

in increasing order. The execution speed of the program should be linear in N.