Uitwerking opgave 1 + opgave 6, werkcollege 6 (2015)

Opgave 1.

```
void langste(int A[ ], int n, int vanaf, int deelrij[ ],
             int lengte, int & max){
   int i;
   for ( i=vanaf; i<n; i++ ){
       if ( deelrij[lengte] < A[i] ) { // consistent: nog steeds stijgend
           deelrij[lengte+1] = A[i];   // uitbreiden dus
           if ( lengte+1 > max )       // langste deelrij tot nu toe?
               max = lengte+1;
           langste(A, n, i+1, deelrij, lengte+1, max);
       }
   }
}
```

Aanroep: langste(A, n, 0, deelrij, 0);
De eerste aanroep gaat goed omdat (i) `deelrij[0]=0`, (ii) de $A[i]$ allemaal groter dan 0 zijn en (iii) de echte deelrij pas op positie 1 begint.

Opgave 6.
a. Als we twee negatieve (< 0) getallen bij elkaar optellen is het antwoord zeker < 0. Als we twee positieve (> 0) getallen bij elkaar optellen is het antwoord > 0. Beide gevallen leveren dus nooit $= 0$ op. Ergo: van alle paren (i, j) met i en j beide oneven of i en j beide even weten we zeker dat $A[i] + A[j] \neq 0$.
b. Brute force: alle paren (i, j) met $i < j$ aflopen en testen of $A[i] + A[j] = 0$. Met inachtneming van de restrictie dat i en j niet beide even of beide oneven zijn.

```
int teller = 0;
for (i = 0; i < n; i++)
  for (j = i+1; j < n; j+=2)
  // nu worden bij elke even index i alleen oneven indices
  // j > i afgelopen, en analoog voor oneven i
    if (A[i] + A[j] == 0)
      teller++;
return teller;
```

c. We splitsen het array nu (herhaald, want recursie) in twee stukken. We tellen het aantal gevraagde paren in de linkerhelft (recursieve aanroep) en in de rechterhelft (recursieve aanroep). Vervolgens moeten we alleen nog paren indices (i, j) controleren waarbij i in de linkerhelft zit en j in de rechterhelft. Wederom onder de restrictie dat we alleen paren (i, j) bekijken met i even en j oneven, of met i oneven en j even.
Merk op dat het basisgeval wordt gegeven door `rechts` = `links` + 1 als we het stuk $A[$`links`$], \ldots, A[$`rechts`$]$ bekijken. Eerste aanroep: aantal = aantal2(A,0,n-1);

```
int aantal2(int A[], int links, int rechts) {
  int teller = 0;
  int m, i, j;
  if (rechts == links+1) { // 2 elementen
    if (A[links]+A[rechts] == 0)
      return 1;
    else
      return 0;
  } // basisgeval twee elementen
  else {
  // meer dan twee elementen: recursieve aanroepen
    m = (links+rechts)/2; // m is altijd oneven
    // aantal paren links en aantal paren rechts optellen
    teller = aantal2(A, links, m) + aantal2(A, m+1, rechts);
    // en nu linkerhelft met rechterhelft vergelijken
    for (i = links; i < m; i++) { // even i links
      for (j = m+2; j <= rechts; j+=2)  // oneven j rechts
        if (A[i] + A[j] == 0)
          teller++;
    }
    for (i = links+1; i <= m; i++) { // oneven i links
      for (j = m+1; j < rechts; j+=2)  // even j rechts
        if (A[i] + A[j] == 0)
          teller++;
    }
    return teller;
  } // else meer dan 2 elementen
} // aantal2
```