Uitwerking tentamen Programmeermethoden 7 maart 2003

```
1.a. bool ispiek (int A[ ], int i) {
       return ( i == 0 || A[i] > A[i-1] ) && ( i == n-1 || A[i] > A[i+1] );
     }  // ispiek
  b. int pieken (int A[ ], int n) {
       int i, tel = 0;
       for ( i = 0; i < n; i++ ) if ( ispiek( A,i) ) tel++;
       return tel;
     }  // pieken
  c. int verschil (int A[ ], int n) {
       int i, j, kl; bool first = true;
       for ( i = 0; i < n-1; i++ ) for ( j = i+1; j < n; j++ )
         if ( ispiek (A,i) && ispiek (A,j) ) {
           if ( first ) {
             first = false;
             if ( A[i] > A[j] ) kl = A[i]-A[j] else kl = A[j]-A[i];
           }  // if
           if ( A[i] > A[j] && A[i]-A[j] < kl ) kl = A[i]-A[j];
           if ( A[i] < A[j] && A[j]-A[i] < kl ) kl = A[j]-A[i];
         }  // if
       return kl;
     }  // verschil
  d. int dalen (int A[ ], int n) {
       int aantal = pieken (A,n) - 1;
       if ( ! ispiek (A,0) ) aantal++;
       if ( ! ispiek (A,n-1) ) aantal++;
       return aantal;
     }  // dalen
  e. bool gesorteerd (int A[ ], int n) {
       return ( pieken (A,n) == 1 && dalen (A,n) == 1 );
     }  // gesorteerd

2.b. Jules 10 6 2  Jules 10 8 2  Jules 0 10 2  Wallace 0 10 2   4  0 10 2
  c. 10 keer:  Jules 10 6 2; daarna:  Wallace 10 4 2  11  10 4 2
  d. Steeds wordt in jules de bovengrens van de for-loop uit wallace
     (keer, gekoppeld aan t) opgehoofd, zodat de loop niet eindigt.
     In feite wordt keer eens te groot, en dan dus negatief, en stopt
     het weer wel.
  e. Ja, dat mag, mits de twee eerste parameters van wallace (keer en x)
     call by value zijn (zonder & dus). Het is geen recursie, tenzij de
     regel in de functie wallace zelf staat.

3.a. bool vorige (char bord[M][N], int x, int y,
                  int & xvorig, int & yvorig ) {
       xvorig = x; yvorig = y;
       if ( bord[x][y] == 'Q' ) return false;
       else if ( bord[x][y] == '<' ) yvorig--;
       else if ( bord[x][y] == '>' ) yvorig++;
       else if ( bord[x][y] == '^' ) xvorig--;
       else if ( bord[x][y] == 'V' ) xvorig++;
       else return false;  // kan eigenlijk niet voorkomen
       return true;
     }  // vorige
  b. bool volgende (char bord[M][N], int x, int y,
                    int & xvolg, int & yvolg ) {
       xvolg = x; yvolg = y;
       if ( y > 0 && bord[x][y-1] == '>' ) {
         yvolg--; return true; }  // if
       if ( y+1 < N && bord[x][y+1] == '<' ) {
         yvolg++; return true; }  // if
       if ( x > 0 && bord[x-1][y] == 'V' ) {
         xvolg--; return true; }  // if
       if ( x+1 < M && bord[x+1][y] == '^' ) {
         xvolg++; return true; }  // if
       return false;
     }  // volgende
  c. void staart (char bord[M][N], int x, int y,
                  int & xuit, int & yuit ) {
       while ( volgende (x,y,xuit,yuit) ) {
         x = xuit; y = yuit;
       }  // while
     }  // staart
```

```
  d. bool kruip (char bord[M][N], int & koprij, int & kopkol) {
       int r = random ( );
       int x = koprij; int y = kopkol; bool groei = false;
       if ( r == 0 )
         if ( y > 0 && ( bord[x][y-1] == ' ' || bord[x][y-1] == '*' ) {
           bord[x][y] = '<'; kopkol--;
           if ( bord[koprij][kopkol] == '*' ) groei = true;
           bord[koprij][kopkol] = 'Q';
         }  // if
         else { cout << "Slang is dood" << endl; return false; }  // else
       else ... analoog 3 keer, voor r == 1/2/3
       if ( !groei ) {
         staart (bord,koprij,kopkol,x,y); bord[x][y] = ' ';
       }  // if
       return true;
     }  // kruip

4.a. void maak (CL & cl) {
       cl.P = new circ_lijst_elem; cl.C = new circ_lijst_elem;
       cl.P->volgende = cl.C; cl.C->volgende = cl.P;
       cl.P->data = 0; cl.C->data = 0;  // hoeft niet
     }  // maak
  b. void CL::voeg_elem_toe ( ) {
       circ_lijst_elem * nieuw = new circ_lijst_elem;
       nieuw->volgende = P->volgende; P->volgende = nieuw;
     }  // voeg_elem_toe
  c. int main ( ) {
       CL cl; int i, tel = 0;
       maak (cl);
       for ( i = 1; i <= 10; i++ )
         cl.voeg_elem_toe ( );
       while ( true ) {
         if ( random ( ) % 2 == 0 ) cl.consumeer ( );
         else {
           cl.produceer (tel); tel++;
         }  // else
       }  // while
       return 0;
     }  // main
  d. int CL::produceer (int info) {
       if ( P->volgende != C ) {
         P->data = info; P = P->volgende; return 1;
       }  // if
       return -1;
     }  // produceer
     int CL::consumeer ( ) {
       if ( C->volgende != P ) {
         C = C->volgende; return C->data;
       }  // if
       return -1;
     }  // consumeer
```