Uitwerking tentamen Programmeermethoden 31 juli 2006

OPGAVE 1

```
a. bool iseen (int A [ ], int n) {
     int i;
     for ( i = 0; i < n-1; i++ ) if ( A[i] != 0 ) return false;
     return ( A[n-1] == 1 );
   }//iseen
b. void kopie (int A[ ], int B[ ], int n) {
     int i; for ( i = 0; i < n; i++ ) A[i] = B[i];
   }//kopie
c. bool telop (int A[ ], int B[ ], int n) {
     int i;
     for ( i = n-1; i >= 0; i-- ) {
       A[i] = A[i] + B[i];
       if ( A[i] > 9 ) {
         A[i] -= 10; if ( i > 0 ) A[i-1]++; else return false;
       }//if
     }//for
     return true;
   }//telop
d. void min1 (int A[ ], int n) {
     int i = n-1; while ( A[i] == 0 ) { A[i] = 9; i--; }//while
     A[i]--;
   }//min1
e. bool maal (int A[ ], int B[ ], int n) {
     int C[n]; bool result; kopie (C,A,n);
     while ( ! iseen (B,n) ) {
       result = telop (A,C,n); if ( ! result ) return false;
       min1 (B,n);
     }//while
     return true;
   }//maal
```

OPGAVE 2

a. Globale variabelen gelden in het gehele programma, en worden helemaal bovenin aangemaakt. Locale variabelen gelden (tijdelijk) alleen in de functie waarin ze aangemaakt zijn.
Variabelen kunnen call by value en call by reference worden meegegeven aan een functie. Bij call by value gaat alleen de waarde van de parameter naar de functie, alwaar een locale variabele deze waarde opvangt, en er met deze locale variabele wordt verder gerekend. De oorspronkelijk variabele behoudt zijn waarde. Bij call by reference (&) gaat als het ware de variabele zelf naar de functie, en kan dan ook blijvend veranderd worden. Eigenlijk wordt het adres (de reference) doorgegeven.
Formeel: in functieheading, bijvoorbeeld x, y in `int f (int x, bool y) {`
Actueel: bij aanroep, bijvoorbeeld r en y in `z = f (r,y);`

b. 6,4,4 6,4,3 6,4,2 6,4,1 1,6,4 50 5,6,4

c. 6,4,4 4,4,5 4,5,3 4,5,3 39 4,5,3

d. 4,4,4 3,3,3 2,2,2 2,2,2 27 2,6,4

e. Dan moet boven bend nog een prototype van snap worden gezet:
`int snap (int a, int b, int c);`
Let er op dat de recursie een keer afbreekt = stopt, het basisgeval dus.

OPGAVE 3

```
a. int leeg (int S[n][n]) {
     int tel = 0, i, j;
     for ( i = 0; i < n; i++ ) for ( j = 0; j < n; j++ )
       if ( S[i][j] == 0 ) tel++;
     return tel;
   }//leeg
b. bool okee (int S[n][n], int i) {
     int A[n+1]; int j;
     for ( j = 0; j < n+1; j++ ) A[j] = 0;
     for ( j = 0; j < n; j++ ) {
       A[S[i][j]]++; if ( A[S[i][j]] > 1 ) return false;
     }//for
     return ( A[0] == 0 );
   }//okee
c. int invulbaar (int S[n][n], int i, int j, int & k) {
     int p, k1, tel = 0; bool ok; k = 0;
     for ( k1 = n; k1 > 0; k1-- ) {
       ok = true;
```

```
      for ( p = 0; p < n; p++ ) if ( p != j && S[i][p] == k1 ) ok = false;
      for ( p = 0; p < n; p++ ) if ( p != i && S[p][j] == k1 ) ok = false;
      if ( ok ) { k = k1; tel++; }//if
    }//for
    return tel;
  }//invulbaar
d. bool invullen (int S[n][n]) {
    int i, j;
    for ( i = 0; i < n; i++ ) for ( j = 0; j < n; j++ )
      if ( S[i][j] == 0  && invulbaar (S,i,j,k) > 0 )
        S[i][j] = k;
    return ( leeg (S) == 0 );
  }//invullen

OPGAVE 4
a. void verwijder (mens* & ingang) {
    mens* weg = ingang;
    if ( ingang != NULL && ingang->leeftijd >= 18 ) {
      ingang = ingang->volgende; delete weg;
    }//if
  }//verwijder
b. void voegtoe (mens* & ingang, int lt) {
    mens* nieuw = new mens;
    nieuw->volgende = ingang; nieuw->leeftijd = lt;
    nieuw->aantal = hoeveel (ingang,lt); ingang = nieuw;
  }//voegtoe
c. void wissel (mens* ingang) {
    int temp;
    if ( ingang != NULL && ingang->volgende != NULL
         && ingang->leeftijd != ingang->volgende->leeftijd ) {
      temp = ingang->leeftijd;
      ingang->leeftijd = ingang->volgende->leeftijd;
      ingang->volgende->leeftijd = temp;
      temp = ingang->aantal;
      ingang->aantal = ingang->volgende->aantal;
      ingang->volgende->aantal = temp;
    }//if
  }//wissel
d. Bij a en b moet er een & bij: de pointer ingang gaat veranderen. (Bij
  a natuurlijk niet altijd, overigens.) Bij c mag het, de pointer ingang
  verandert daar toch niet.
e. int hoeveel (mens* ingang, int lt) {
    mens* loper = ingang;
    while ( loper != NULL ) {
      if ( loper->leeftijd == lt ) return 1 + loper->aantal;
      loper = loper->volgende;
    }//while
    return 0;
  }//hoeveel
```