Uitwerkingen Tentamen Programmeermethoden 9 januari 2006

OPGAVE 1.

```
a.void gk (double B[ ], int & gr, int & kl, int n) {
    int i; gr = 0; kl = 0;
    for ( i = 1; i < n; i++ ) {
      if ( B[i] > B[gr] ) gr = i;
      if ( B[i] < B[kl] ) kl = i;
    }//for
  }//gk
b.int stijgdaal (double B[ ], int n) {
    int i, tel = 2;  // aanname: n >= 2
    for ( i = 1; i < n-1; i++ )
      if  ( ( B[i-1] < B[i] && B[i] > B[i+1] ) ||
            ( B[i-1] > B[i] && B[i] < B[i+1] ) )
        tel++;
    return tel;
  }//stijgdaal
c.bool opso (double B[ ], int n) {
    int kl, gr; gk (B,gr,kl,n);
    return ( stijgdaal (B,n) == 2 && kl == 0 );
  }//opso
d.int lang (double B[ ], int n) {
   int lengte = 1, grlengte = 1, i;
   for ( i = 1; i < n; i++ ) {
     if ( B[i] > B[i-1] ) lengte++;
     else lengte = 1;
     if ( lengte > grlengte ) grlengte = lengte;
   }//for
   return grlengte;
 }//lang
e.bool opso2 (double B[ ], int n) {
    return ( lang (B,n) == n );
  }//opso2
```

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.78, 3 en 3 13 en 2 15, 3 en 4
c.78, 3 en 7 7 en 3 10, 7 en 3
d.995, 995 en 995 1003 en 1003 2006, 3 en 8

e.Dan moet boven rita nog een prototype van salomon worden gezet:

```
    int salomon (int x, int y);
```

Let er op dat de recursie een keer afbreekt = stopt, het basisgeval dus.

OPGAVE 3.

```
a.void positief (int numbers[ ][n]) {
    int i, j;
    for ( i = 0; i < m; i++ ) for ( j = 0; j < n; j++ )
      if ( numbers[i][j] < 0 ) numbers[i][j] = -numbers[i][j];
  }//positief
b.int kolom (int numbers[ ][n], getal) {
    int i, j, som;
    for ( j = 0; j < n; j++ ) {
      som = 0;
      for ( i = 0; i < m; i++ ) som += numbers[i][j];
      if ( som == getal ) return j;
    }//for
    return -1;
  }//kolom
c.int achterelkaar (int numbers[ ][n], waarde) {
    int i, j, tel = 0;
    for ( i = 0; i < m; i++ ) {
```

```
      j = 0;
      while ( j < n-1 ) {
        if ( numbers[i][j] == numbers[i][j+1] && waarde == numbers[i][j] ) {
          tel++;
          j = n;  // stop de while loop, een break kan ook
        }//if
        j++;
      }//while
    }//for
    return tel;
  }//achterelkaar
d.void lopen (int numbers[ ][n], int & rij, int & kolom) {
    bool doorgaan = true; rij = 0; kolom = 0;
    while ( doorgaan ) {
      if ( rij < m-1 && kolom < n-1 &&
           numbers[rij+1][kolom] == numbers[rij][kolom+1] ) {
        rij++; kolom++;
      }//if
      else if ( rij < m-1 && kolom > 0 &&
           numbers[rij+1][kolom] == numbers[rij][kolom-1] ) {
        rij++; kolom--;
      }//if
      else doorgaan = false;
    }//while
  }//lopen

OPGAVE 4.
a.void voegtoe (informatie* & ingang, int nieuw) {
    informatie* denieuwe = new informatie;
    denieuwe->jaartal = nieuw; denieuwe->volgende = ingang;
    if ( ingang != NULL ) denieuwe->vervolgende = ingang->volgende;
    else denieuwe->vervolgende = NULL;
    ingang = denieuwe;
  }//voegtoe
b.void verwijder (informatie* & ingang) {
    informatie* weg = ingang;
    if ( ingang != NULL && ingang->jaartal != 2006 ) {
      ingang = ingang->volgende; delete weg;
    }//if
  }//verwijder
c.void verwissel (informatie* ingang) {
    int temp;
    if ( ingang != NULL && ingang->vervolgende != NULL ) {
      temp = ingang->jaartal;
      ingang->jaartal = ingang->vervolgende->jaartal;
      ingang->vervolgende->jaartal = temp;
    }//if
  }//verwissel
d.Bij a en b moet het, bij c mag het. Bij a en b gaat de pointer ingang
  wijzigen (bij b niet als hij al NULL was, overigens). Bij c verandert
  deze pointer toch niet.
e.int lengte (informatie* ingang) {
    int tel = 0; informatie* loper = ingang;
    while ( loper != NULL ) {
      if ( loper->vervolgende != NULL ) {
        loper = loper->vervolgende;
        tel += 2;
      }//if
      else {
        tel++;
        if ( loper->volgende != NULL ) tel++;
        loper = NULL;
      }//else
    }//while
    return tel;
  }//lengte
```