Uitwerking Tentamen Programmeermethoden 10 januari 2005

OPGAVE 1

```
a.void bubblesort (int A[ ], int n) {
    int ronde, i, temp;
    for ( ronde = 1; ronde < n; ronde++ )
      for ( i = 0; i < n-ronde; i++ )
        if ( A[i] > A[i+1] ) {
          temp = A[i]; A[i] = A[i+1]; A[i+1] = temp; }//if
  }//bubblesort
b.int verschillend ( int A[ ], int n) {
    int aantal = 1, i;
    for ( i = 0; i < n-1; i++ )
      if ( A[i] != A[i+1] ) aantal++;
    return aantal;
  }//verschillend
c.int hoevaak (int A[ ], int n, int ind, int waarde) {
    int tel = 0, i = ind;
    while ( i < n && waarde > A[i] ) i++;
    while ( i < n && A[i] == waarde ) { i++; tel++; }//while
    return tel;
  }//hoevaak
d.void vaakst (int A[ ], int n, int & meest, int & zovaak) {
    zovaak = 0;
    for ( i = 0; i < n; i++ )
      if ( hoevaak (A,n,i,A[i]) >= zovaak ) {
        meest = A[i]; zovaak = hoevaak (A,n,i,A[i]); }//if
    // je kunt ook steeds hoevaak (A,n,i,A[i]) verder gaan in het array
  }//vaakst
```

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.merijn: 6,10,52,26
frodo: 5,26,52

c.merijn: 6,40,62,52
frodo: 40,52,62

d.merijn: 6,4,26,18 respectievelijk merijn: 6,4,19,18
frodo: 4,18,26 frodo: 4,18,19
varianten wegens volgorde evaluaties bij + (en dan veranderde q) pepijn in q = pepijn (q,p) + pepijn (q,y) + x;

e.Mag, mits er een prototype void merijn (int,int&); boven de definitie van pepijn wordt gezet. Het is recursie, want de functies roepen elkaar dan (indirect) aan.

OPGAVE 3

```
a.int aantal (int Puz1[m][n], int X) {
    int tel = 0, i, j;
    for ( i = 0; i < m; i++ )
      for ( j = 0; j < n; j++ )
        if ( Puz1[i][j] == X ) tel++;
    return tel;
  }//aantal
b.bool perm (int Puz1[m][n], int Puz2[m][n]) {
```

```
    int X;
    for ( X = 0; X <= 20; X++ )
      if ( aantal (Puz1,X) != aantal (Puz2,X) )
        return false;
    return true;
  }//perm
c.int groto (int Puz1[m][n], int Puz2[m][n]) {
    int i, j, k, l, gro = 0, tel; bool okee;
    for ( i = 0; i < m; i++ )
      for ( j = 0; j < n; j++ ) {
        tel = 0; k = i; l = j; okee = true;
        while ( okee ) {
          if ( k == m || l == n || Puz1[k][l] != Puz2[k+1][l+1] )
            okee = false;
          else {
            tel++; k++; l++;
            if ( k == m || l == n || Puz2[k][l] != Puz1[k+1][l+1] )
              okee = false;
            else { tel++; k++; l++; }//else
          }//else
        }//while
        if ( tel > gro ) gro = tel;
      }//for
    return gro;
  }// groto

OPGAVE 4
a.void voegtoe (getal* & begin, int X) {
    getal* nieuw = new getal;
    nieuw->volgende = begin; begin = nieuw; begin->inhoud = NULL;
    if ( X == 1 ) {
      nieuw = new getal;  // OF hulp->inhoud = new getal; etcetera
      nieuw->volgende = NULL; nieuw->inhoud = NULL; begin->inhoud = nieuw;
    }//if
  }//voegtoe
b.void gooiweg (getal* & begin) {
    getal* weg = begin;
    if ( begin != NULL ) {
      begin = begin->volgende;
      if ( weg->inhoud != NULL )
        delete weg->inhoud;  // formeel hoeft de test niet ...
      delete weg;
    }//if
  }//gooiweg
c.void hoogop (getal* begin) {
    getal* nieuw = new getal; nieuw->inhoud = NULL;
    nieuw->volgende = begin->inhoud; begin->inhoud = nieuw;
  }//hoogop
d.int som (getal* begin) {
    int tel = 0; getal* boven = begin; getal* onder;
    while ( boven != NULL ) {
      onder = boven->inhoud;
      while ( onder != NULL ) {
        tel++; onder = onder->volgende; }//while
      boven = boven->volgende;
    }//while
    return tel;
  }//som
e.Bij a en b moet er een & bij (de begin van de lijst kan gaan
  veranderen), bij c(!) en d hoeft het niet.
```