UITWERKINGEN Tentamen Programmeermethoden 7 januari 2008

OPGAVE 1

```
a. int cijfers (int A[ ], int i) {
      int tel = 1, get = A[i];
      while ( get >= 10 ) { get = get / 10; tel++; }//while
      return tel;
   }//cijfers
b. int bijelkaar (int A[ ], int x, int vanaf) {
     int plek = vanaf, temp, i;
     for ( i = vanaf; i < n; i++ )
       if ( cijfers (A,i) == x ) {
         temp = A[i]; A[i] = A[plek]; A[plek] = temp; plek++; }//if
     return plek-vanaf;
   }//bijelkaar
c. void allemaal (int A[ ]) {
     int vanaf = 0, decijfers = 1;
     while ( vanaf < n ) {  // of zelfs < n-1
       vanaf += bijelkaar (A,decijfers,vanaf); decijfers++; }//while
  }//allemaal
d. void sorteer (int A[ ]) {
     int i, ronde, 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
   }//sorteer
```

e. Methode van d: n(n-1)/2 vergelijkingen.
En die van c: enigszins afhankelijk van de lengte van de voorkomende getallen. Stel maximale lengte t (bijvoorbeeld t=8), dan t aanroepen van bijelkaar. Dan ruwweg n operaties (inclusief lengte getallen), dus ongeveer tn werk. Dus is c beter. De methode van d doet meer dan nodig.

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. 2, 3 en 2008   2, 6 en 2008   1, 3 en 2
c. 1, 1 en 8   1, 6 en 8   1, 1 en 1
d. Ze roepen elkaar steeds weer aan ... oneindige loop.
e. Mag, mits de parameters van arnon en afth call by value zijn.

OPGAVE 3

```
a. bool controle (int nurikabe[ ][n]) {
     int i, j;
     for ( i = 0; i < m; i++ ) for ( j = 0; j < n; j++ )
       if ( ( nurikabe[i][j] > 0 && i+1 < m && nurikabe[i+1][j] > 0
              && nurikabe[i][j] != nurikabe[i+1][j] )
            || ( nurikabe[i][j] > 0 && j+1 < n && nurikabe[i][j+1] > 0
               && nurikabe[i]IEEE Transactions on Systems, Man, and Cybernetics[j] != nurikabe[i][j+1] ) )
         return false;
     return true;
   }//controle
b. void meer (int nurikabe[ ][n], int i, int j) {
     int aantal, k, l; if ( nurikabe[i][j] != 0 ) return;
     nurikabe[i][j] = -1;
     for ( aantal = 0; aantal < m*n; aantal++ )
       for ( k = 0; k < m; k++ ) for ( l = 0; l < n; l++ )
         if ( nurikabe[k][l] == -1 ) {
           if ( k > 0 && nurikabe[k-1][l] == 0 ) nurikabe[k-1][l] = -1;
           if ( l > 0 && nurikabe[k][l-1] == 0 ) nurikabe[k][l-1] = -1;
           if ( k+1 < m && nurikabe[k+1][l] == 0 ) nurikabe[k+1][l] = -1;
           if ( l+1 < n && nurikabe[k][l+1] == 0 ) nurikabe[k][l+1] = -1;
         }//if
   }//meer
c. int hoeveelmeer (int nurikabe[ ][n]) {
     int i, j, tel = 0;
     for ( i = 0; i < m; i++ ) for ( j = 0; j < n; j++ )
       if ( nurikabe[i][j] == 0 ) { meer (nurikabe,i,j); tel++; }//if
     for ( i = 0; i < m; i++ ) for ( j = 0; j < n; j++ )
       if ( nurikabe[i][j] == -1 ) nurikabe[i][j] = 0;
     return tel;
   }//hoeveelmeer
```

OPGAVE 4

```
a. void verwissel (info* ingang) {
     if ( ingang != NULL && ingang->rechts != NULL
          && ingang->let > ingang->rechts->let ) {
       char temp = ingang->let; ingang->let = ingang->rechts->let;
```

```
      ingang->rechts->let = temp; }//if
  }//verwissel
b. void voegtoe (info* & ingang,char let,int get) {
     info* nieuw = new info; nieuw->rechts = ingang;
     nieuw->letter = let; nieuw->aantal = get; ingang = nieuw;
     if ( get == 1 ) {
       nieuw = new info; nieuw->rechts = NULL; nieuw->aantal = 0;
       nieuw->letter = (char)((int)let + 1 );
       nieuw->boven = NULL; ingang->boven = nieuw;
     }//if
     else ingang->boven = NULL;
  }//voegtoe
c. void verwijder (info* & ingang) {
     info* weg = ingang;
     if ( ingang != NULL ) {
       ingang = ingang->rechts;
       if ( weg->boven != NULL )  // of: if ( weg->aantal > 0 )
         delete weg->boven;
       delete weg;
     }//if
  }//verwijder
d. Bij b en c moet het erbij, omdat de ingangspointer gaat veranderen
   (tenzij bij c hij al NULL was, trouwens). Bij a hoeft het niet, de
   pointer ingang verandert toch niet; het maakt daar niet uit.
e. int desom (info* ingang) {
     int res = 0; info* loper = ingang; info* erboven;
     while ( loper != NULL ) {
       erboven = loper;
       while ( erboven != NULL ) {
         res += erboven->aantal; erboven = erboven->boven; }//while
       loper = loper->rechts; }//while
     return res;
  }//desom
```