Uitwerkingen Tentamen Programmeermethoden 28 maart 2008

OPGAVE 1

```
a. int komtvoora (int A[ ], int x) {
     int i, teller = 0;
     for ( i = 0; i < n; i++ ) if ( A[i] == x ) teller++;
     return teller;
   }//komtvoora
b. int komtvoorb (int A[ ], int x) {
     int i, teller = 0, waar;
     for ( i = 0; i < n; i++ )
       if ( A[i] == x ) { teller++; waar = i; } //if
     if ( teller != 1 ) return teller;
     else return -waar-1;
   }//komtvoorb
c. bool verschillend (int A[ ]) {
     int i;
     for ( i = 0; i < n-1; i++ )
       if ( komtvoorb (A,A[i]) > 0 ) return false;
     return true;
   }//verschillend
d. int grootste (int A[ ]) {
     int i = 0, teller = 0;
     while ( teller != n ) { teller += komtvoora (A,i); i++; }//while
     return i-1;
   }//grootste
e. void sorteer (int A[ ]) {
     int i, kl, dekl, temp;
     for ( i = 0; i < n; i++ ) { dekl = i;
       for ( kl = i+1; kl < n; kl++ ) if ( A[kl] < A[i] ) dekl = kl;
       temp = A[i]; A[i] = A[dekl]; A[dekl] = temp;
     }//for
   }//sorteer
```

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. 3,7,4,17   3,7,4,17   2,4,3,1   1,4,3
c. 3,7,4,17   3,10,4,17 2,10,3,1   1,10,2
d. Oneindige loop in de functie paul, omdat de bovengrens van de loop steeds wordt opgehoogd
e. Bij paul (y, z): z wordt z+y, y blijft onveranderd
   En david (y, z): z wordt met 1 afgelaagd, y wordt opgehoogd met z * ceiling (x/2) (waarbij ceiling naar boven afrondt)

OPGAVE 3

```
a. bool controle (char puzzel[ ][n], int i) {
     int j;
     for ( j = 0; j < n/2; j++ )
       if ( puzzel[i][j] != puzzel[i][n-1-j] ) return false;
     return true;
   }//controle
b. int telze (char puzzel[ ][n]) {
     int i, j, teller = 0;
     for ( i = 1; i < m; i++ ) for ( j = 0; j < n; j++ )
       if ( ( puzzel[i][j] == 'A' || puzzel[i][j] == 'E' ||
            puzzel[i][j] == 'I' || puzzel[i][j] == 'O' ||
            puzzel[i][j] == 'U' ) && ( puzzel[i-1][j] == 'A' ||
            puzzel[i-1][j] == 'E' || puzzel[i-1][j] == 'I' ||
            puzzel[i-1][j] == 'O' || puzzel[i-1][j] == 'U' ) )
         teller++;
     return teller;
```

```
   }//telze
c. bool ja (char puzzel[ ][n], char woord[ ], int w) {
     int i, j, i1, j1, index;
     for ( i = 0; i < m; i++ ) for ( j = 0; j < n; j++ ) {
       if ( woord[0] == puzzel[i][j] ) {
         index = 0; i1 = i; j1 = j;
         while ( index < w-1 ) {
           index++;
           if ( i1 > 0 && puzzel[i1-1][j1] == woord[index] ) i1--;
           else if ( j1 > 0 && puzzel[i1][j1-1] == woord[index] ) j1--;
           else if ( i1 < m-1 && puzzel[i1+1][j1] == woord[index] ) i1++;
           else if ( j1 < n-1 && puzzel[i1][j1+1] == woord[index] ) j1++;
           else index = w+10;
         }//while
         if ( index == w-1 ) return true;
       }//if
     }//for
     return false;
   }//telze

OPGAVE 4
a. void voegtoe (info* & ingang, char let, int get) {
     info* nieuw = new info; nieuw->letter = let;
     nieuw->getal = get; nieuw->volg = ingang; nieuw->terug = NULL;
     if ( ingang != NULL && ingang->volg != NULL )
       ingang->volg->terug = nieuw;
     ingang = nieuw;
   }//voegtoe
b. void verwijder (info* & ingang) {
     info* weg = ingang;
     if ( ingang != NULL &&
          ingang->letter != 'Q' && ingang->getal != 8 ) {
       ingang = ingang->volg; delete weg;
       if ( ingang != NULL && ingang->volg != NULL )
         ingang->volg->terug = NULL;
     }//if
   }//verwijder
c. void verwissel (info* & ingang) {
     info* twee;
     if ( ingang != NULL && ingang->volg != NULL
          && ingang->get > ingang->volg->get ) {
       twee = ingang->volg; ingang->volg = twee->volg; ingang = twee;
     }//if
   }//verwissel
d. Bij a, b en c moet het erbij, omdat de ingangspointer gaat veranderen
   (tenzij bij b hij al NULL was of ..., trouwens).
e. info* voorste (info* pijl) {
     while ( pijl->terug != NULL ) pijl = pijl->terug;
     if ( pijl->volg->terug != NULL ) pijl = pijl->volg->terug;
     return pijl;
   }//voorste
```