UITWERKING Tentamen Programmeermethoden 4 januari 2013

OPGAVE 1

a.
```
void klein (int A[ ], int i, int & kl, int n) {
     int j; kl = i;
     for ( j = i+1; j < n; j++ )
       if ( A[j] < A[kl] ) kl = j;
   }//klein
```

b.
```
bool gesorteerd (int A[ ], int i, int n) {
     int j;
     for ( j = i; j < n-1; j++ )
       if ( A[j] > A[j+1] ) return false;
     return true;
   }//gesorteerd
```

c.
```
void sorteer (int A[ ], int n) {
     int i = 0, kl, temp;
     while ( ! gesorteerd (A,i,n) ) {
       klein (A,i,kl,n);
       temp = A[i]; A[i] = A[kl]; A[kl] = temp; i++; }//while
   }//sorteer
```

d. Een maal de functie gesorteerd doen (n-1 vergelijkingen), en dan klaar.
Het gaat dus om een al gesorteerd rijtje, en alleen dan gebeurt dit.

e. De functie gesorteerd doet n-i-1 vergelijkingen; en klein ook.
Maximaal 2 (n-1 + n-2 + ... + 2 + 1) = n (n-1).
Dit is te realiseren door een oplopend gesorteerd rijtje, waarbij alleen het laatste getal verkeerd staat.

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. M1,3 LF M3,3 LF 2,3 LF M3,2 LF 1,3 LF M3,1 LF 0,3 LF 5,3,11
(waarbij LF staat voor een regelovergang)

c. M1,3 LF M1,5 LF 0,5 LF 17,5,10

d. M0,0 LF 1,2,11

e. Dit mag alleen als er geen & staat voor de eerste parameter van diederik,
en er boven mark een prototype van diederik wordt gezet.

OPGAVE 3

a.
```
int duos (bool T[ ][m]) {
     int i, j, tel = 0;
     for ( i = 0; i < m; i++ ) for ( j = i+1; j < m; j++ )
       if ( T[i][j] && T[j][i] ) tel++;
     return tel;
   }//duos
```

b.
```
int druk (bool T[ ][m]) {
     int drukstation = 0, tel, drukte = -1, i, j;
```

```
     for ( i = 0; i < m; i++ ) {
       tel = 0;
       for ( j = 0; j < m; j++ ) { // eventueel if ( i != j ) ...
         if ( T[i][j] ) tel++;
         if ( T[j][i] ) tel++; }//for
       if ( tel >= drukte ) {
         drukstation = i;
         drukte = tel;
       }//if
     }//for
     return drukstation;
   }//druk
c. bool bereik (bool T[ ][m], int i, int j) {
     int k;
     for ( k = 0; k < m; k++ )
       if ( T[i][k] && T[k][j] ) return true;
     return false;
   }//bereik
d. int aantal (T[ ][m], int i) {
     int k = i+1, tel = 1;
     while ( k < m ) {
       if ( bereik (T,i,k) ) { i = k; tel++; }//if
       k++;
     }//while
     return tel;
   }//aantal

OPGAVE 4
a. void verwijder (element* & begin) {
     element* weg = begin;
     if ( begin != NULL ) {
       if ( begin->onder != NULL ) delete begin->onder;
       begin = begin->rechts; delete weg;
     }//if
   }//verwijder
b. void voegtoe (element* & begin, int getal) {
     element* nieuw;
     if ( begin == NULL || begin->info % 2 == 0 ) {
       nieuw = new element;
       nieuw->info = getal; nieuw->onder = NULL;
       nieuw->rechts = begin; begin = nieuw;
     }//if
   }//voegtoe
c. void voegtoe2 (element* begin, int getal) {
     if ( begin != NULL && begin->onder == NULL ) {
       begin->onder = new element; begin->onder->info = getal;
       begin->onder->onder = NULL; begin->onder->rechts = NULL;
     }//if
   }//voegtoe2
d. Bij a en b moet er een & bij, want de ingangspointer kan gaan
veranderen.
   Bij c hoeft het niet, de ingangspointer verandert niet. Het mag hier
wel.
e. void zetrechts (element* begin) {
     element* p = begin; element* q = NULL;
     while ( p != NULL ) {
       if ( p->onder != NULL ) {
         if ( q != NULL ) q->rechts = p->onder;
         q = p->onder;
       }//if
       p = p->rechts;
```

```
  }//while
  if ( q != NULL ) q->rechts = NULL;
}//zetrechts
```