Uitwerking tentamen Programmeermethoden 31 maart 2006

OPGAVE 1

a.
```
void bubblesort (double A[ ], int n) {
    int i, j; double temp;
    for ( i = 1; i < n; i++ )
      for ( j = 0; j < n-i; j++ )
        if ( A[j] > A[j+1] ) {
          temp = A[j]; A[j] = A[j+1]; A[j+1] = temp; }//if
  }//bubblesort
```

b.
```
void herstel1 (double A[ ], int n) {
    int i = 0; bool klaar = false;
    while ( ! klaar ) {       // zoek de boosdoener
      if ( A[i] > A[i+1] ) {  // A[i+1] is het
        A[i+1] *= 2; klaar = true;  // of return;
      }//if
      i++;
    }//while
  }//herstel1
```

c.
```
void herstel2 (double A[ ], int n) {
    int i = 0; double temp;
    for ( i = n-1; i > 0; i-- )
      if ( A[i] < A[i-1] ) {
        temp = A[i]; A[i] = A[i-1]; A[i-1] = temp; }//if
  }//herstel2
```

d.
```
void herstel3 (double A[ ], int n) {
    int i;
    for ( i = 0; i < n-1; i++ )  // zoek de boosdoener
      if ( 2*A[i] < A[i+1] ) {   // A[i] is het
        A[i] *= 2; return;
      }//if
  }//herstel3
```

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. 35, 8, 36 28, 6, 2037 4, 3 en 4

c. 35, 8, 36 32, 11, 2041 4, 3 en 4

d. 13, 15, 7 13, 17, 42 13, 17 en 42

e. Een prototype `void twee (int a, int b, int c);` boven de declaratie van "een" toevoegen. Let er op dat de recursie stopt, doordat er een basisgeval is.

OPGAVE 3.

a.
```
int vier (int C[n][n]) {
    int tel = 0, i, j, b;
    for ( i = 0; i < n-1; i++ )
      for ( j = 0; j < n-1; j++ )
        for ( b = 1; i+b < n && j+b < n; b++ )
          if ( C[i][j] == C[i][j+b] && C[i][j] == C[i+b][j]
               && C[i][j] == C[i+b][j+b] ) tel++;
    return tel;
  }//vier
```

b.
```
void krimp (int C[n][n], int & i, int & j, int & b) {
    int hoogi = i, hoogj = j;
    if ( C[i+b][j] > C[hoogi][hoogj] ) { hoogi = i+b; hoogj = j; }//if
    if ( C[i][j+b] > C[hoogi][hoogj] ) { hoogi = i; hoogj = j+b; }//if
    if ( C[i+b][j+b] > C[hoogi][hoogj] ) { hoogi = i+b; hoogj = j+b; }//if
    b--;
    if ( hoogi != i ) i++;
    if ( hoogj != j ) j++;
  }//krimp
```

c.
```
int pers (C[n][n]) {
```

```
    int tel = 0, i = 0, j = 0, b = n-1, ioud, joud;
    while ( b > 0 ) {
      ioud = i; joud = j;
      krimp (C,i,j,b); tel++;
      if ( ioud == i && joud == j ) return tel;
    }//while
    return tel;
  }//pers

OPGAVE 4
a.void voegtoe (info* & ingang) {
    info* nieuw = new info;
    nieuw->andere = NULL; nieuw->aantal = 0;
    nieuw->volgende = ingang; ingang = nieuw;
  }//voegtoe
b.void verwijder (info* & ingang) {
    info* weg = ingang;
    if ( ingang != NULL ) {
      if ( ingang->andere != NULL ) // of "aantal" testen, of zonder test
        delete ingang->andere;
      ingang = ingang->volgende;
      delete weg;
    }//if
  }//verwijder
c.void vulaan (info* ingang, int getal) {
    info* nieuw;
    if ( ingang != NULL && ingang->aantal == 0 ) {
      nieuw = new vakje;
      nieuw->andere = NULL;
      nieuw->volgende = NULL;
      nieuw->aantal = getal;
      ingang->andere = nieuw;
      ingang->aantal = 1;
    }//if
  }//vulaan
d.Bij a en b moet er een & bij, want de ingangspointer gaat veranderen
  (bij b overigens niet als hij al NULL was), bij c mag het - maar hoeft
  het niet, de ingangspointer verandert toch niet.
e.int lengte (info* ingang) {
    int tel = 0; info* loper = ingang;
    while ( loper != NULL ) {
      tel += loper->aantal + 1;
      loper = loper->volgende;
    }//while
    return tel;
  }//lengte
```