Uitwerking Tentamen Programmeermethoden 6 augustus 2007

OPGAVE 1

```
a. bool run (int A[ ], int i, int j) {
     int k;
     for ( k = i; k < j; k++ )
       if ( A[i] != A[i+1] + 1 && A[i] != A[i+1] - 1 ) return false;
     return true;
   }//run
b. int longrun (int A[ ], int n) {
     int maxlen = 1, i, j;
     for ( i = 0; i < n; i++ ) for ( j = i+1; j < n; j++ )
       if ( run (A,i,j) && j - i + 1 > maxlen ) maxlen = j - i + 1;
     return maxlen;
   }//longrun
c. int longrun2 (int A[ ], int n) {
     int maxlen = 1, i, curlen = 1;
     for ( i = 1; i < n; i++ )
       if ( A[i-1] == A[i] + 1 || A[i-1] == A[i] - 1 ) {
         curlen++; if ( curlen > maxlen ) maxlen = curlen; }//if
       else curlen = 1;
     return maxlen;
   }//longrun2
d. void bubblesort (int A[ ], int n) {
     int temp, i, ronde;
     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
```

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. 1(true) 12 6   1 12 12   12 12 12 1
c. 1 12 6   0(false) 7 12   7 12 3 1
d. 1 6 6   1 6 6 6 6 1
e. Ja, mits de variabele x in gerard maar call by value is. Het is namelijk geen "l-value", iets wat links mag staan bij een toekenning.

OPGAVE 3

```
a. int hoogste (int hoogte[m][ ], int & i, int & j) {
     int hoog = 0, p, q;
     for ( p = 0; p < m; p++ ) for ( q = 0; q < n; q++ )
       if ( hoogte[p][q] >= hoog ) {
         hoog = hoogte[p][q]; i = p; j = q; }//if
     return hoog;
   }//hoogste
b. bool kan (int A[m][ ], int i, int j, int r, int s) {
     int dx = 0, dy = 0;
     if ( i == r ) // zelfde rij
       if ( j < s ) dy = 1; else dy = -1;
     else if ( j == s ) // zelfde kolom
       if ( i < r ) dx = 1; else dx = -1;
     else return false;
     while ( i != r || j != s )
       if ( hoogte[i][j] >= hoogte[i+dx][j+dy] ) {
         i += dx; j += dy; }//if
       else return false;
     return true;
   }//kan
c. int tel (int hoogte[m][ ]) {
```

```
    int i, j, r, s, telze = 1; int dummy = hoogste (hoogte,i,j);
    bool bereikt[m][n]; bool changes = true;
    for ( r = 0; r < m; r++ ) for ( s = 0; s < n; s++ )
      bereikt[r][s] = false;
    bereikt[i][j] = true;
    while ( changes ) {
      for ( i = 0; i < m; i++ ) for ( j = 0; j < n; j++ )
        for ( r = 0; r < m; r++ ) for ( s = 0; s < n; s++ )
          if ( bereikt[i][j] && ! bereikt[r][s]
               && kan (hoogte,i,j,r,s) ) {
            bereikt[r][s] = true; telze++; changes = true; }//if
    }//while
    return telze;
  }//tel
```

OPGAVE 4

a.
```
  void voegtoe (info* & ingang, int get) {
    info* nieuw = new info; info->getal = get;
    info->hori = ingang; ingang = nieuw;
    nieuw = new info; nieuw-getal = get;
    nieuw->vert = ingang; ingang->vert = nieuw;
    if ( ingang->hori == NULL ) nieuw->hori = NULL;
    else nieuw->hori = ingang->hori->vert;
  }//voegtoe
```

b.
```
  void verwijder (info* & ingang) {
    info* weg;
    if ( ingang != NULL && ingang->getal % 2 == 0 ) {
      weg = ingang; ingang = ingang-> hori;
      delete weg->vert; delete weg; }//if
  }//verwijder
```

c.
```
  void verwissel (info* ingang) {
    int temp;
    if ( ingang != NULL && ingang->hori != NULL ) {
      temp = ingang->getal;
      ingang->getal = ingang->hori->getal;
      ingang->hori->getal = temp;
      temp = ingang->vert->getal;  // hoeft niet
      ingang->vert->getal = ingang->vert->hori>getal;
      ingang->vert->hori->getal = temp; }//if
  }//verwissel
```

d. Bij a en b moet het (de ingangspointer gaat veranderen; bij b niet altijd overigens). Bij c hoeft het niet, de ingangspointer blijft onveranderd.

e.
```
  bool check (info* ingang) {
    info* loop = ingang;
    while ( loop != NULL )
      if ( loop->getal != loop->vert->getal ) return false;
      else loop = loop->hori;
    return true;
  }//check
```