Iteratieve oplossing in C:

```
int fac(int n)
{
        int i, result;

        result = 1;
        for(i = 1; i <= n; i++)
                result *= i;
        return result;
}
```

5. Schrijf een functie **void max (int n, int * m)** die **n** positieve getallen in de invoer leest en het grootste getal teruggeeft in **m**.

Oplossing in C:

```
void max(int n, int *m)
{
        int i, getal;

        *m = 0;
        for (i=0; i<n; i++)
        {
                getal = getint();
                if (getal > *m)
                        *m = getal;
        }
}
```

6. Schrijf een functie **void fibon (int n, int r[])**, die de eerste **n** Fibonacci getallen berekent en bewaart in rij **r** van lengte **n**. Schrijf tevens een hoofdprogramma dat **m** inleest en de eerste **m** Fibonacci-getallen berekent en afdrukt.

Oplossing in C:

```
void fibon(int n, int r[])
{
        int i;

        r[0] = 1;
        r[1] = 1;
        for (i=2; i<n; i++)
                r[i] = r[i-1] + r[i-2];
}

int m, i, f[100];
main ()
{
        m = getint();
        fibon(m, f); /* bereken fibonacci getallen */
        for(i=0; i<m; i++)
                printint(f[i]); /* druk fibonacci getallen */
}
```

Alternatieve definitie voor de functie:

```
void fibon(int n, int *r)
{
     int i;

     *r++ = 1;
     *r++ = 1;
     for (i=2; i<n; i++)
     {
          *r = *(r-1) + *(r-2);
          r++;
     }
}
```

7. Schrijf een programma dat een rij van tien getallen inleest en ze daarna in omgekeerde volgorde afdrukt. Schrijf functies voor het inlezen en afdrukken van de rij.

Eerste oplossing in C:

```
void lees(int r[])
{
     int i;

     for (i=0; i< 10; i++)
          r[i] = getint();
}

void schrijfOmgekeerd(int r[])
{
     int i;

     for (i=9; i>= 0; i--)
          printint(r[i]);
}
```

Tweede oplossing in C:

```
void lees(int *r)
{
     int i;

     for (i=0; i< 10; i++)
          *r++ = getint();
}

void schrijfOmgekeerd(int *r)
{
     int i;

     r += 10;
     for (i=0; i< 10; i++)
          printint(*(--r));
}
```

8. Schrijf een programma dat twee rijen **a** en **b** van tien getallen inleest. Bereken vervolgens een nieuwe rij **c** waarin elk element de som is van de twee overeenkomstige elementen in **a** en **b**. Druk daarna de berekende rij af. Herbruik de resultaten van de vorige oefening!

Oplossing in C:

```
void som(int *r1, int *r2, int *r3)
{
     int i;

     for (i=0; i<10; i++)
          *r3++ = *r1++ + *r2++;
}

void schrijf(int *r)
{
     int i;

     for (i=0; i< 10; i++)
          printint(*r++);
}

int a[10], b[10], c[10];

main()
{
     lees(a);
     lees(b);
     som(a,b,c);
     schrijf(c);
}
```

9. Schrijf een recursieve functie die nagaat of een rij van tien elementen gespiegeld is rond het middelste element.

Oplossing in C:

```
int spiegelRec(int *r, int begin, int end)
{
     if (begin >= end)
          return 1;
     else
          if (*(r + begin) == *(r + end))
               return spiegelRec(r, begin+1, end-1);
          else
               return 0;
}

int spiegel(int *r)
{
     return spiegelRec(r, 0, 9);
}
```

10. Schrijf een programma dat de som bepaalt van de kolommaxima van een matrix. Gebruik een functie die het maximum bepaalt van één kolom. Aan de invoer komt **n**, **m** en de **nxm**-matrix (rij per rij). Schrijf een functie voor het inlezen van de matrix.

Oplossing in C:

```
int maxkolom (int t[][10], int n, int k)
{
     int i, max;

     max = t[0][k];
     for (i=1; i<n; i++)
          if (max < t[i][k])
               max = t[i][k];
     return (max);
}

void leesmatrix (int t[][10], int *ar, int *ak)
{
     int i, j;

     *ar = getint();
     *ak = getint();
     for (i=0; i < *ar; i++)
          for (j=0; j < *ak; j++)
               t[i][j] = getint();
}

int mat[10][10];

main()
{
     int n, m, som, j;

     leesmatrix (mat, &n, &m);
     som = 0;
     for (j=0; j<m; j++)
          som += maxkolom (mat, n, j);
     printint(som);
}
```

11. Schrijf een functie die het maximum van de kolommaxima van een matrix bepaalt. Gebruik hiervoor de functie uit de vorige opgave.

Oplossing in C:

```
int maxkolmaxima (int t[][10], int ar, int ak)
{
     int j, max, h;

     max = maxkolom (t, ar, 0);
     for (j=1; j<ak; j++)
     {
```

```
            h = maxkolom(t, ar, j);
            if (h>max)
            max = h;
        }
        return(max);
}
```

12. Schrijf een recursieve functie die het probleem van de 'torens van Hanoi' oplost voor maximaal tien schijven. De functie moet telkens afdrukken hoe een schijf moet verplaatst worden. Het aantal schijven wordt ingelezen. De stapels worden genummerd van 1 tot 3.

Voorbeeld:

Invoer:

```
3
```

Uitvoer:

```
1 3
1 2
3 2
1 3
...
1 3
```

Oplossing in C:

```
int hanoi(int n, int van, int naar)
{
    int via;

    /* deze oplossing geeft alleen de uit te voeren
       verplaatsingen van schijven aan */

    via = 6 - van - naar;
    if (n == 0)
        return 1;

    /* verplaats n-1 schijven van 'van' naar 'via' */
    hanoi(n-1, van, via);

    /* verplaats een schijf van 'van' naar 'naar' */
    printint(van, naar);

    /* verplaats n-1 schijven van 'via' naar 'naar' */
    hanoi(n-1, via, naar);
}

main()
{
    int aantal;

    aantal = getint();
    hanoi(aantal, 1, 3);
}
```

Oefeningen

1. Schrijf een programma dat rationale getallen inleest en terug afdrukt tot een ongeldig rationaal getal (noemer = 0) ingegeven wordt. Definieer een record-type voor de rationale getallen bestaande uit 3 velden: geheel deel, teller en noemer.

Oplossing in C:

```
struct rationaalgetal
{
     int teller;
     int noemer;
     int gehdeel;
};

void leesrationaal(register struct rationaalgetal *g)
{
     g->gehdeel = getint();
     g->teller = getint();
     g->noemer = getint();
}

void drukrationaal(struct rationaalgetal g)
{
     printint(g.gehdeel,g.teller,g.noemer);
}

main()
{
     struct rationaalgetal q;

     leesrationaal(&q);
     while(0 != q.noemer)
     {
          drukrationaal(q);
          leesrationaal(&q);
     }
}
```

2. Pas de vorige oefening aan zodat de rationale getallen vereenvoudigd worden vooraleer ze afgedrukt worden. Maak gebruik van de functies uit de oefeningen van sectie 8.5.

Oplossing in C:

```
int ggd(int a,int b)
{
     if (b == 0)
          return a;
     return ggd(b, a % b);
}
```

```
void vereenvoudig(struct rationaalgetal in,
                  struct rationaalgetal *uit)
{
     int deler;

     deler = ggd(in.teller, in.noemer);
     in.teller = in.teller / deler;
     in.noemer = in.noemer / deler;
     uit->gehdeel = (in.teller / in.noemer) + in.gehdeel;
     uit->teller = in.teller % in.noemer;
     uit->noemer = in.noemer;
}

main()
{
     struct rationaalgetal q,r;

     leesrationaal(&q);
     while(q.noemer != 0)
     {
          vereenvoudig(q, &r);
          drukrationaal(r);
          leesrationaal(&q);
     }
}
```

3. Pas de vorige oefening aan zodat alle verschillende rationale getallen (in vereenvoudigde vorm) in een rij weggeborgen worden. De invoer bevat 100 rationale getallen. Er kunnen dubbels voorkomen.

Oplossing in C:

```
int isgelijk(struct rationaalgetal a,
             struct rationaalgetal b)
{
     ...
}

void drukrij(struct rationaalgetal *r, int n)
{
     ...
}

struct rationaalgetal rij[100];

main()
{
     struct rationaalgetal q;
     int i,j,in;

     in = 0;
     leesrationaal(&(rij[in]));
```

```
        for(i = 1; i < 100; i++)
        {
                leesrationaal(&q);
                vereenvoudig(q, &(rij[++in]));
                j = 0;
                while (!isgelijk(rij[j],rij[in]))
                        j++;
                if (j != in)
                        in--;
        }
        drukrij(rij, ++in);
}
```

4. Pas de vorige oefening aan zodat de rij gesorteerd is van klein naar groot.

Oplossing in C:

```
/*
 * pre: a en b zijn reeds vereenvoudigd
 * post: 1 --> (a < b) / 0 --> a >= b
 */

int iskleiner(struct rationaalgetal a,
              struct rationaalgetal b)
{
        if (a.gehdeel < b.gehdeel)
                return 1;
        if (a.gehdeel > b.gehdeel)
                return 0;
        return((a.teller*b.noemer) < (b.teller*a.noemer));
}

struct rationaalgetal rij[100];

main()
{
        struct rationaalgetal q;
        int i,j,t,in;

        in = 0;
        leesrationaal(&(rij[in]));
        for(i = 1; i < 100; i++)
        {
                leesrationaal(&q);
                vereenvoudig(q,&(rij[++in]));
                j = 0;
                while (iskleiner(rij[j],rij[in]))
                        j++;
                if (j != in)
                {
                        if (isgelijk(rij[j],rij[in]))
```

```
                    in--;
               else
               {
                    q = rij[in];
                    for (t = in ; t > j; t--)
                         rij[t] = rij[t-1];
                    rij[j] = q;
               }
          }
     }
     drukrij(rij, in + 1);
}
```

5. Gegeven een rij van carthesische coördinaten; zoek het punt dat het verst van de oorsprong ligt. De coördinaten worden ingelezen.

Oplossing in C:

```
struct Tco
{
     int x, y;
};

struct Tco corij[100];

int cogr (struct Tco co1, struct Tco co2)
{
     if (co1.x * co1.x + co1.y * co1.y >
                    co2.x * co2.x + co2.y * co2.y)
          return (1);
     else
          return (0)
}

void leesco (register struct Tco *co)
{
     co->x = getint():
     co->y = getint():
}

void schrijfco (struct Tco co)
{
     printint (co.x, co.y);
}

void leescorij (struct Tco cr[], int n)
{
     int i;
     for (i = 0; i < n; i++)
          leesco (&cr[i]);
}
```

```
void vindmaxco(struct Tco cr[],
               struct Tco *maxco, int n)
{
     int i, max;

     max = 0;
     for (i = 1; i < n; i++)
     {
          if (cogr (cr[i], cr[max]) == 1)
               max = i;
     }
     *maxco = cr[max];
}

main()
{
     struct Tco maxco;
     int len;

     len = getint();
     leescorij (corij, len);
     vindmaxco (corij, &maxco, len);
     schrijfco (maxco);
}
```

6. Schrijf een pakket van routines om te werken met getalstrings van variabele lengte. In onderstaande oplossing is aangenomen dat de resultaatstring steeds kan bewaard worden in de beschikbare rij.

Oplossing in C:

```
struct stringtype
{
     int lengte;
     int getallen[80];
};


void leesstring (struct stringtype *s)
{
     int i;

     printstr("lengte: ");
     s->lengte = getint();

     printstr("getallen: ");
     for (i=0; i<s->lengte; i++)
          s->getallen[i] = getint();
}
```

```
void drukstring (struct stringtype s)
{
     int i;
     for (i=0; i<s.lengte; i++)
          printint (s.getallen[i]);
}

void copystring(struct stringtype *s,
                    struct stringtype ct)
{
     int i;

     s->lengte = ct.lengte;
     for (i=0; i<ct.lengte; i++)
          s->getallen[i] = ct.getallen[i];
}

void concatstring(struct stringtype *s,
                    struct stringtype ct)
{
     int i;
     for(i=0; i<ct.lengte; i++)
          s->getallen[s->lengte+i] = ct.getallen[i];
     s->lengte += ct.lengte;
}

int stringvgl(struct stringtype cs,
               struct stringtype ct)
{
     int i;

     if (cs.lengte <> ct.lengte)
          return 0;
     else
     {
          i = 0;
          while((i < cs.lengte) &&
                    (cs.getallen[i] == ct.getallen[i]))
               i++;
          if (i == cs.lengte) return 1;
          else return 0;
     }
}
```

Welke wijzigingen zou je aanbrengen aan bovenstaande definities omwille van efficiëntieredenen?

7. Gebruik de routines uit vorige oefening in het volgende programma:

```
struct stringtype s, t;

main()
{
     leesstring (&s);
     leesstring (t);

     printstr("s == t? ");
     printint (stringvgl(s,t));

     printstr("s = ");
     drukstring (s);

     printstr("t = ");
     drukstring (t);

     copystring (&s,t);

     printstr("s = ");
     drukstring (s);

     printstr("t = ");
     drukstring (t);
}
```