Oefenzitting 3 - Functies

Oefeningen sectie 1.10 - Functies in Drama

1. Teken de activatie-records voor alle functies en stel de toekenningstabellen op. Vertaal daarna de code en teken de stapel zoals hij eruit ziet op het ogenblik van de toekenning aan totaal.

```
int bereken (register int van, int naar, int categorie, int via,
register int kost, register int verblijf)
{
   register int korting, leeftijd, totaal;
   int bon, conditie, vervoer;

   ...
   totaal = categorie * kost - korting - bon;
   ...
   return(totaal + vervoer)

}

void voorstel (int afstand, register int continent, register int
land1, int land2)
{
   register int klasse, luxe;
   int geld, bezit;

   ...
   geld = bezit + 2 * afstand - bereken(land1, land2, klasse,
continent, luxe, 10);
   printint (geld);
   ...
}

main ()
{
   register int bestemming, werelddeel;
   int vertrek;

   ...
   voorstel (5000, werelddeel, vertrek, bestemming);
}
```

2. Vertaal het volgende C-programma dat het kleinst gemene veelvoud van twee getallen berekent. Het maakt gebruik van een andere functie die de grootst gemene deler berekent. Teken de stapel zoals ze er uit ziet net voor de terugkeer uit de functie ggd.

```
int ggd(register int a, int b)  // grootst gemene deler
{
   register int r;

   r = a % b;
   while (r != 0)
   {
       a = b;
       b = r;
       r = a % b;
   }
```

```
   return b;
}

int kgv(int a, register int b)  // kleinst gemeen veelvoud
{
   int gv;
   gv = ggd(a, b);
   return (a*b / gv);
}

int r[10];

main ()
{
     register int i;
     int g[9];

     r[0] = getint();
     for (i = 1; i < 10; i++) {
       r[i] = getint();
       g[i-1] = kgv(r[i], r[i-1]);
       printint(r[i-1], r[i], g[i-1]);
     }
}
```

3. Vertaal het volgende C-programma.
   Teken de stapel zoals ze eruit ziet net na de berekening van b*b.

```
void kwadraat(register int a, int b, register int * skw, int *vkw)
{
    int aa, bb;
    aa = a * a;
    bb = b * b;
    *skw = aa + bb;
    *vkw = aa - bb;
}

main ()
{
   int x;
   register int y;
   int som;
   int verschil;

   x = 5; y = 3;
   kwadraat (x, y+4, &som, &verschil);
   printint (som, verschil);
}
```

4. Recursieve functie
   - Vertaal het volgende C-programma. De recursieve functie 'fibon' berekent het *n*-de Fibonacci-getal.
   - Teken de stapel zoals ze er uitziet wanneer fibon(4) opgeroepen is vanuit main, en de stapel het hoogst gevuld is, net voor de teruggave van de waarde 1.
   - Voer dit programma uit en geef als invoer 5, en een tweede keer 15. Waarom duurt het zo lang om dit getal te berekenen?
   - Hoeveel keer wordt de functie fibon opgeroepen wanneer n=4? Veralgemeen de formule. Is dit een goede implementatie om een Fibonacci-getal te berekenen?

```
int fibon (int n)
{
    if (n == 0 || n == 1)
      return (1);
    else
      return fibon(n-1) + fibon(n-2);
}

main()
{
   int g;
   g = getint();
   printint (g, fibon(g));
}
```

5. Bestudeer het volgende Drama-programma. Voor elk bevel of constante staat tussen haakjes het geheugenadres waarin deze zal opgeslagen worden.
   - Welk getal wordt afgedrukt?
   - Hoeveel bevelen worden er uitgevoerd?
   - Hoe vaak wordt er iets uit het geheugen gelezen?
   - Hoe vaak wordt er iets in het geheugen geschreven?
     - Hints:
       - Een bevel moet eerst opgehaald worden (=lezen) om uitgevoerd te kunnen worden
       - Iets op de stapel zetten = schrijven
       - Iets van de stapel halen = lezen

```
(0000:)        HIA.w R1,3
(0001:)        HIA   R0,TABEL(R1)
(0002:)        OPT.w R0,1
(0003:)        BST   R0
(0004:) LUS:   VGL.w R1,0
(0005:)        VSP   GEL,DRK
(0006:)        AFT.w R1,1
(0007:)        KTG
(0008:) DRK:   DRU
(0009:)        STP
(0010:) TABEL: 3
(0011:)        2
(0012:)        1
(0013:)        0
(0014:)        -1
(0015:)        -2
       EINDPR
```