TU Delft Faculteit EWI

Tentamen Programmeren in C (EE1400)

11 april 2011, 9.00 – 12.00

Uitwerkingen

Opgave 1 (24 punten)

a. `9 15`

b. `4 8`

c. `6 7`

d. `30 11`

e. `6 8`

f. `h F`

Opgave 2 (8 punten)

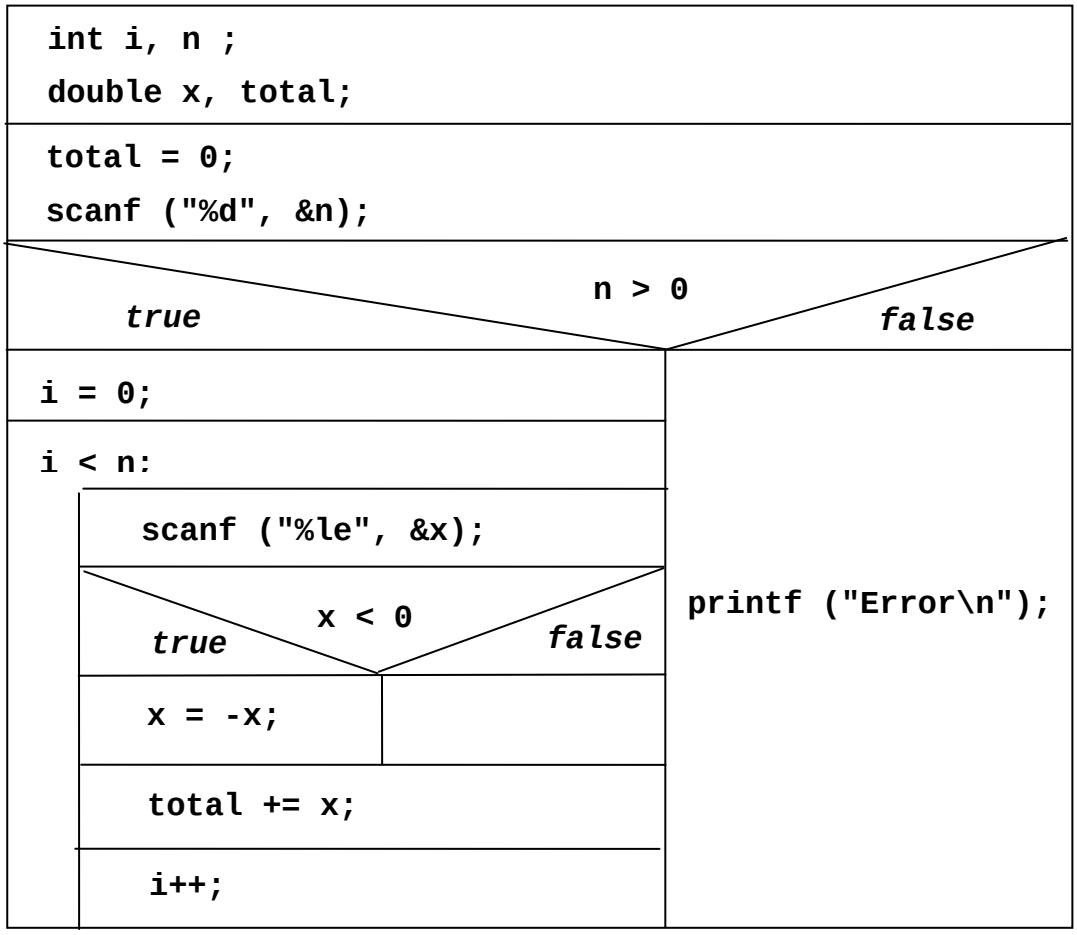
a. `16`

b. `7`

c. `8`

d. `24`

Opgave 3 (10 punten)



Opgave 4 (8 punten)

```
void concatstrings (char *s[], char buf[])
{
     int i, j;
     int k = 0;

     for (i = 0; s[i]; i++) {
          for (j = 0; s[i][j] != '\0'; j++)
               buf[k++] = s[i][j];
     }
     buf[k] = '\0';
}
```

Opgave 5 (8 punten)

```
double integral_f (double a, double b, int N)
{
     extern double f(double x);
     int i;
     double val = 0.0;
     double delta = (b - a)/N;

     for (i = 1; i <= N ; i++) {
          val += f(a + i * delta);
     }
     val *= delta;
     return val;
}
```

Opgave 6 (8 punten)

.

```
#define  ABS(x) (x < 0 ? (-x) : (x))

int diagonal_dominant(double a[N][N])
{
     int i, j;
     double sum;

     for (i = 0 ; i < N; i++) {
          sum = 0;
          for (j = 0 ; j < N ; j++) {
               if (j != i) sum += ABS (a[i][j]);
          }
          if (sum > ABS (a[i][i]))
               return 0;
     }
     return 1;
}
```

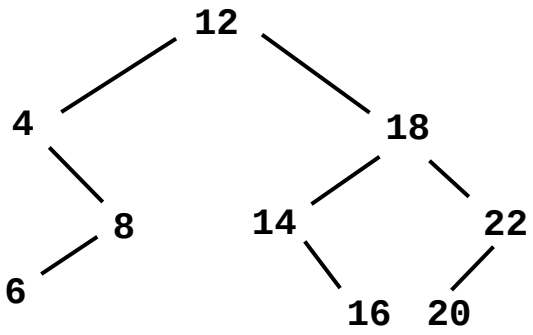
Opgave 7 (8 punten)

```
#include <stdio.h>       /* voor printf() */
#include <string.h>      /* voor strcmp() */

void print_address (char *person_name)
{
     int i;
     for (i = 0; i < nr_members; i++) {
          if (strcmp (members[i].name, person_name) == 0) {
               printf ("%s %d\n", members[i].street,
                                  members[i].number);
               break;
          }
     }
}
```

Opgave 8 (16 punten)

a.



b.

Een rechter subtree moet eerst worden geprint (hogere waarden), dan de node zelf, en tenslotte de linker subtree nodes (lagere waarden).

```
void print_high_to_low (struct node * n)
{
     if (n -> right)
          print_high_to_low (n -> right);

     printf ("%d ", n -> data);

     if (n -> left)
          print_high_to_low (n -> left);
}
```

c.

Met recursie bepalen we de diepte van de linker en rechter subtree en geven dan de grootste waarde door. Als er geen subtrees zijn is de return waarde 0.

```
int tree_depth (struct node * n)
{
     int dl, dr;

     if (n -> left)
          dl = tree_depth (n -> left);
     else
          dl = -1;

     if (n -> right)
          dr = tree_depth (n ->right);
     else
          dr = -1;

     if (dl > dr) return (1 + dl);
     else return (1 + dr);
}
```