Übung vom 10.06.98 bis 16.06.98

Aufgabe 17:

Was berechnet das folgende Programmstück(führen Sie einen Trace durch)?

```
#include <stdio.h>

int a  =0;
int sum=0;

int f(int a)
{
    int b=0;
    if(a>0)
        while(a) b+=a--;
    sum += b;
    return b;
}

int main()
{
    int b;
    printf("Bitte Endwert eingeben: ");
    scanf("%d",&b);
    if(b>0)
        for(a=1;a<=b;a++)
            printf("\tf(%d) = %d\n",a,f(a));
    printf("sum : %d\n", sum);
    return 0;
}
```

Aufgabe 18:

In Aufgabe 17 liefert main() einen Return-Wert. Wer verarbeitet diesen und wofür kann er verwendet werden?

Aufgabe 19:

Schreiben Sie eine rekursive Funktion, die die n-te Fibonacci-Zahl berechnet. Die Fibbonacci-Zahlen sind durch folgende rekurrente Formel definiert:

$$fib(0) = 0$$

$$fib(1) = 1$$

$$fib(n) = fib(n-1) + fib(n-2), \qquad \text{für } n \geq 2$$

Testen Sie die Funktion, indem Sie in main() wiederholt vom Benutzer Werte eingeben lassen und die entsprechnede Fibonacci-Zahl berechnen und ausgeben. Überlegen Sie sich ein geeignetes Abbruchkriterium.

Aufgabe 20:(Abgabe bis Fr. 19.06.98, 17 Uhr)

(Lassen Sie sich nicht vom Umfang der Aufgabe erschrecken; die Lösung ist kürzer als die Aufgabenstellung)

Binomialkoeffizienten – Pascalsches Dreieck

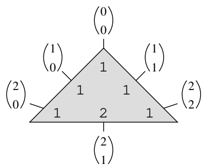
Die Binomialkoeffizienten $\binom{n}{k}$ lassen sich durch folgende rekursive Berechnungsvorschrift ermitteln:

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

$$\binom{0}{0} = 1$$

$$\binom{n}{k} = 0 \qquad \text{für} \begin{cases} k > n \\ k < 0 \\ n < 0 \end{cases}$$

Diese Berechnungsvorschrift spiegelt sich im Pascalschen Dreieck wider: In Zeile n an k-ter Stelle steht der Wert des Binomialkoeffizienten $\binom{n}{k}$, der sich aus der Addition der beiden darüberliegenden Werte ergibt:



a) Schreiben Sie eine Funktion, die **rekursiv** – analog der oben angegebenen Formel – den Binomialkoeffizienten $\binom{n}{k}$ berechnet.

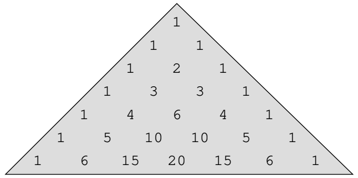
b) Die Binomialfunktion aus a) soll in einem eigenen Modul liegen. Erstellen Sie eine zugehörige Header-Datei, die den Funktionsprototypen (= Funktionsdeklaration) der Binomialfunktion enthält.

c) Implementieren Sie – in einer eigenen Datei – ein kleines Programm zum Testen des Binomialmoduls. Das Testprogramm soll wiederholt Werte für n und k einlesen, $\binom{n}{k}$ berechnen und das Ergebnis ausgeben. Überlegen Sie sich ein geeignetes Abbruchkriterium.

d) Um die Binomialfunktion effizienter zu gestalten, sollen alle bereits berechneten Werte in einem zweidimensionalen Array abgelegt werden, so daß nachfolgende Aufrufe auf bereits berechnete Werte zurückgreifen können. Erweitern Sie das Binomialmodul entsprechend um ein Array, das nur im Modul global bekannt sein soll. Die Größe des Arrays soll durch eine Präprozessor-Konstante in der zugehörigen Header-Datei festgelegt werden.

Hinweis: Erinnern Sie sich an den Zusammenhang von Binomialkoeffizienten und dem Pascalschen Dreieck und speichern Sie berechnete Werte an der entsprechenden Stelle.

Pascalsches-Dreieck

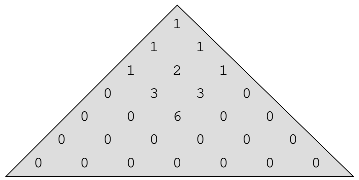


als 2-dim. Array

1						
1	1					
1	2	1				
1	3	3	1			
1	4	6	4	1		
1	5	10	10	5	1	
1	6	15	20	15	6	1

e) Bevor das Array verwendet wird, sollte es mit Nullen initialisiert werden. Hierfür benötigen Sie eine weitere Funktion im Binomialmodul, die alle Elemente des Arrays auf 0 setzt (vergessen Sie nicht den Prototypen in der Header-Datei einzutragen).

f) Ergänzen Sie das Binomialmodul um eine Ausgabefunktion, die das Array als Pascalsches Dreieck ausgibt. Nach der Berechnung von $\binom{4}{2}$ und Array-Größe 7 sollte die Ausgabe etwa so aussehen:



(Diese Funktion kann für eine etwaige Fehlersuche sehr nützlich sein ...)

g) Setzen Sie die Array-Größe auf 50, kompilieren Sie Ihr Testprogramm erneut und berechnen Sie, wieviele Möglichkeiten es beim Lotto gibt, 6 aus den 49 Kugeln zu ziehen.