Übungsblatt 3

Aufgabe 1

c) Pseudocode binäre Suche

```
funct binäreSucheRekursiv (array, gesuchteZahl, u, o) returns integer
    addiere u und o und teile durch 2, ergibt mitte;
    if (gesuchteZahl == mitte);
        return mitte;
    else if (gesuchteZahl < array[mitte] && u<=o)
            return binäreSucheRekursiv (array, gesuchteZahl, u, mitte-1);
        else if (gesuchteZahl > array[mitte] && u<=o)
                return binäreSucheRekursiv (array, gesuchteZahl, mitte-1, o);
            else
                return -1;
            fi
        fi
    fi
tcnuf
```

Aufgabe 2

Komplexitätsexperiment:

		Größe des Arrays		
	Suchverfahren	1024	2048	4096
erfolgreich	Linear (erster Treffer)	513	942	2122
	Linear (letzter Treffer)	1024	2048	4096
	Binär iterativ	10	11	12
	Binär rekursiv	10	11	12
erfolglos	Linear (erster Treffer)	1024	2048	4096
	Linear (letzter Treffer)	1024	2048	4096
	Binär iterativ	10	11	12
	Binär rekursiv	11	12	13

Auswertung:
- Die Arraygröße spielt vor allem bei der linearen Suche eine entscheidende Rolle, da sich hier die Anzahl der Schlüsselvergleiche um denselben Faktor erhöht, um den sich auch die Größe des Arrays erhöht.
- Bei der binären Suche dagegen ist die Größe des Arrays nahezu unbedeutend. Die Schlüsselvergleiche erhöhen sich bei doppelter Größe gerade mal um 2.

Das Experiment ist nicht nötig. Die Werte lassen sich mit Hilfe folgender Formeln berechnen:

	Anzahl Vergleiche			
Suchmethode	**bester Fall**	**schlechtester Fall**	**Durchschnitt (erfolgreiche Suche)**	**Durchschnitt (erfolglose Suche)**
lineare Suche (erster Treffer)	1	n	≈n/2	n
lineare Suche (letzter Treffer)	n	n	n	n
binäre Suche	1	$\approx \log_2 n$	$\approx \log_2 n$	$\approx \log_2 n$

Begründungen:

lineare Suche (erster Treffer): Diese Suchmethode geht eine Zahlenfolge so lange durch, bis der gesuchte Wert das erste Mal auftaucht. Wenn dies der Fall ist, wird die Schleife abgebrochen.

➔ Der beste Fall, der auftreten kann, ist, dass gleich das erste überprüfte Element die gesuchte Zahl enthält und die Schleife somit nur einmal durchlaufen wird.

➔ Im schlechtesten Fall enthält das letzte Element die gesuchte Zahl, womit alle Elemente durchsucht werden müssen.

➔ Bei der erfolgreichen Suche erhält man, je nach gesuchter Zahl, einen Wert, zwischen 0 und der Anzahl n der Elemente – 1. Bei mehrmaliger Durchführung pendelt sich der Durchschnitt bei n/2 ein. Grund: Gauß'sche Summenformel

➔ Bei der erfolglosen Suche ist der Rückgabewert immer n. Somit ist auch der Durchschnitt n.

lineare Suche (letzter Treffer): Diese Suchmethode geht eine Zahlenfolge *immer* von vorne bis ganz nach hinten durch.

➔ Bei der Suche erhält man als Anzahl der Vergleiche immer die Länge des Arrays zurück.

➔ Bester und schlechtester Fall, sowie der Durchschnitt der erfolgreichen und erfolglosen Suche sind immer gleich (=n)

binäre Suche: Die binäre Suche benötigt eine sortierte Zahlenfolge. Man beginnt, indem man prüft, ob das Element in der Mitte größer oder kleiner als die gesuchte Zahl ist. Je nach dem, fährt man mit der Suche dann in der oberen, bzw. in der unteren Hälfte der Folge mit der Suche fort. Danach wird die Mitte neu berechnet, und man führt erneut Schritt 1 durch. Dies passiert so lange, bis der gesuchte Wert gefunden ist, bzw. bis die untere Grenze größer als die obere Grenze ist (→ Zahl nicht gefunden).

➔ Der beste Fall tritt ein, wenn die erste berechnete Mitte gleich die gesuchte Zahl enthält. Somit ist der gelieferte Wert 1.

➔ Im schlechtesten Fall wird die Zahlenfolge so lange geteilt, bis nur noch ein Element vorhanden ist. Der Wert, den man dann erhält, lässt sich mit der Formel $\log_2 n$ berechnen.

➔ Führt man die Suche jetzt mehrere Male erfolgreich durch, erhält man als Durchschnitt $\log_2 n$ Vergleiche

➔ Da die erfolglose Suche *immer* mit der gleichen Anzahl an Vergleichen abgeschlossen wird, wie der schlechteste Fall, ergibt sich daraus auch ein Durchschnitt von $\log_2 n$.