Aufgabenblatt 3

Aufgabe 1 Rekursive Funktionen

Definieren Sie folgende Funktionen rekursiv:

a) bool istElement(int x, int a[], int n);
prüft, ob x in a vorkommt. n ist die Anzahl der Elemente in a.

b) double max(double a[], int n);
liefert das größte Elemente aus a zurück. n ist die Anzahl der Elemente in a.

c) double mittelwert(double a[], int n)
liefert den Mittelwert aller Elemente aus a zurück. n ist die Anzahl der Elemente in a.

Schreiben Sie ein Hauptprogramm, das die Funktionen ausreichend testet.

Wichtig: Verwenden Sie genau die oben beschriebenen Parameter. Globale oder statische Variablen und Schleifen sind nicht gestattet.

Aufgabe 2 Teile-und-Hersche-Verfahren

a) Stellen Sie fest, was folgende Teile-und-Herrsche-Funktion leistet:

```
bool f(int a[], int l, int r)
{
      if (l >= r)
            return true;
      else
      {
            int m = (l+r)/2;
            return f(a,l,m) && f(a,m+1,r) && a[m] <= a[m+1];
      }
}
```

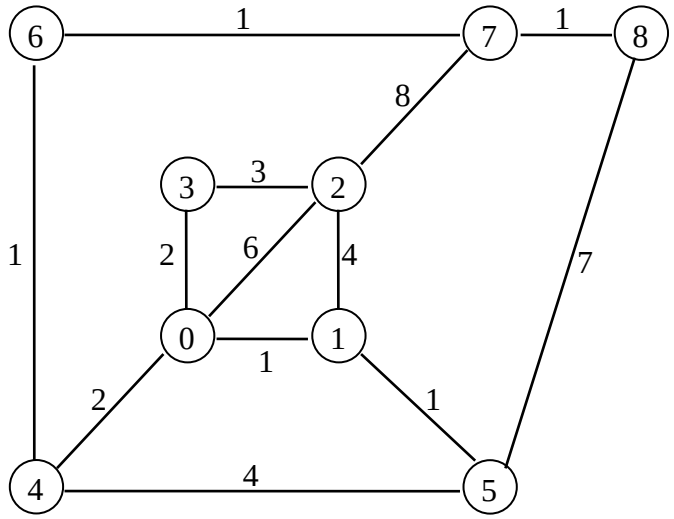
b) Die Funktion soll dann so erweitert werden, dass die Aufrufstruktur textuell auf dem Bildschirm dargestellt wird. Dazu soll bei jedem Aufruf der Funktion die aktuelle Rekursionstiefe und die Parameterwerte ausgegeben werden. Um die Aufrufstruktur deutlicher hervorheben zu können, soll die gewünschte Ausgabe um *2n* Zeichen nach rechts versetzt ausgegeben werden, wobei *n* die aktuelle Rekursionstiefe ist.

Hinweis: arbeiten Sie mit einer lokalen statischen Variablen (`static`), die die aktuelle Rekursionstiefe speichert.

c) Erweitern Sie mit der gleichen Technik wie in Teilaufgabe b) die Funktion so, dass auch noch die Anzahl der Funktionsaufrufe gezählt werden. Geben Sie allgemein an, wieviele Aufrufe maximal stattfinden, wenn n Elemente in a enthalten sind. Bringt die Teile-und-Herrsche-Funktion einen Geschwindigkeitsvorteil gegenüber einer herkömmlichen iterativen Lösung?

Aufgabe 3 Backtracking

Gegeben sei eine Menge von Städten (von 0 bis N-1 nummeriert), die durch Straßen mit einer gewissen Entfernung verbunden sind. In der Abbildung ist ein solches Städtenetz mit den Städten 0 bis 8 zu sehen. Beispielsweise ist Stadt 0 mit Stadt 2 durch eine Strasse der Entfernung 6 verbunden.



Entwickeln Sie eine rekursive Funktion, die durch Backtracking alle Wege von einer Stadt A nach einer Stadt B mit zugehöriger Entfernung generiert, wobei jede Stadt höchstens einmal besucht werden darf.

Beispielsweise gibt es 16 unterschiedliche Wege von 0 nach 8.

Hinweis: Benutzen Sie eine Matrix *benachbart*, wobei

benachbart(i,j)	= *n*,	falls Stadt *i* mit Stadt *j* durch eine Straße der Länge *n* direkt verbunden ist (benachbart ist)
	= -1,	falls Stadt *i* mit *j* nicht direkt verbunden ist.

Besetzen Sie die Matrix mit den in der Abbildung angegebenen Werten.

Abgabe:

Vorführen und Erklären des Programms.