Lösungen:

a) Die Liste wird bis zum Ende durchlaufen. Dort wird das neue Element eingefügt. Der Zeiger des letzten Elementes wird anschließend auf das neue Element gesetzt.

Zeilen 7-15 werden ersetzt durch:

```
p1^.nachfolger:=anker;
anker:=p1;
```

b)

	Name	linker Zeiger	rechter Zeiger
1.	Neumann	2	4
2.	Church	6	3
3.	Gödel		7
4.	Zuse	5	
5.	Turing		8
6.	Babbage		
7.	Hilbert		
8.	Wirth		

c) Baum: Suchalgorithmen einfach und schnell, Einfügen ebenfalls
Indexbäume schnell erstellbar (für andere Sortierungen)
Löschen ggfs aufwendig (falls vollständiges Löschen)
Ab und an Optimierung des Baumes nötig

verkettete Liste: Suchen ggfs aufwendiger, ebenso ggfs das Einfügen
Löschen einfacher

Eine ausreichende Leistung ist gegeben, wenn der Algorithmus in a prinzipiell richtig beschrieben wird und der Baum richtig dargestellt wird. Für eine gute Leistung sollte auch das Einfügen zu Beginn der Liste dargestellt werden sowie die Vorteile. Die Unterschiede zwischen Baum und verketteter Liste sollten dargestellt werden können.

Fragen:

zu a)

Unterschied p und p^ bzw Typ: ^TName

Verketten Sie diese Liste (Reihenfolge nicht ändern) lexikographisch.

Möglicher Algorithmus?

Lösung zur lexikographischen Verkettung:

```
if anker=nil then
      anker:=p1
   else
       begin
       p2:=anker;p3:=anker;
      while ( p2^.nachfolger<>nil) and (p2^.nachname<p1^.nachname) do
         begin
         p3:=p2;
         p2:=p2^.nachfolger;
```

```
        end;
      if (p2^.nachfolger=nil) and (p2^.nachname<p1^.nachname) then
          p2^.nachfolger:=p1
       else
          begin
          p1^.nachfolger:=p2;
          if anker<>p2 then
               p3^.nachfolger:=p1
            else
               anker:=p1;
          end;
     end;
```

zu b)
Wo findet man noch Baumstrukturen (nicht binär)? Xml-Dateien, auch in HTML (s. Tags)

weitere Datenstrukturen: Stapel und Schlange