PGP Gedächtnisprotokoll '22

1. Allgemeines [5P]
   a) Erklären Sie kurz wie die Paradigmen, die in der Vorlesung vorgestellt wurden, sich mit der Programmiersprache C umsetzen lassen? [5 Punkte]

   - Lösungsansatz:
     Prozedural -> man programmiert in C
     Objektorientiert -> man programmiert in C und kann Structs benutzen
     Funktional -> man programmiert ohne Seiteneffekte und kann ggf functionpointer benutzen
     Parallel -> MPI
     Logisch -> Resolutionsalgorithmus implementieren. Fakten aus Datenbank einlesen implementieren.
     Die Antwort „Bibliotheken nutzen" zählt auch (außer bei den Paradigmen, die C auch ohne Bibliotheken automatisch implementiert, wie das Prozedurale Paradigma)

2. Prozedurales Paradigma [18P]
   a) hier hast du ein Programm in C. Zeichne den Stack zu Zeitpunkt x.
   b) Was ist Call-by-Value und Call-by-Reference?

   - Lösungsansatz:
     Call-by-Reference: Als Parameter wird ein Pointer auf den Wert übergeben. Die Funktion kann also den Originalwert verändern.
     Call-by-Value: Als Parameter wird eine Kopie des Wertes übergeben. Der Originalwert bleibt unverändert.

   c) Was ist Fragmentierung? Was kann man dagegen tun?

   - Lösungsansatz:
     Fragmentierung entsteht bei wiederholter nutzung und Freigabe unterschiedlich großer Speicherbereiche. Dabei entstehen freie Speicherbereiche, die zu klein sind, um sinnvoll genutzt werden zu können.
     Dieses Problem lässt sich durch Defragmentierung (den genutznen Speicher lückenlos aneinander schieben, und damit nur noch einen großen freien Speicherbereich erhalten) lösen.
     Es gibt Speicherverwaltete Programmiersprachen, die das für einen übernehmen.

   d) Speicher Auslese Problem in C
   e) Unterschied zwischen Programmiersprachen mit und ohne Speicherverwaltung?

3. Objektorientierung [20P]
   a) zwei Möglichkeiten Objektinstanziierung in Java

   - Lösungsansatz:
     Konstruktor
     Instanzenvariableniniziierung
     Instanzeniniziierung

   b) static vs non static Attribute

   - Lösungsansatz:
     Static Attribute sind Klassenattribute. Die gibt es ein Mal für die ganze Klasse.
     Nicht Static Attribute sind Objektattribute. Jedes Objekt hat dieses Attribut und

kann unterschiedliche Werte annehmen.

c) linearisiere diese Vererbungshierarchie
d) Klassendiagramm Methodenzugriffe mit Packages
e) Die Programmiersprache… aka ist Java Objectorientiert

- Lösungsansatz:
  Java ist Objektorientiert, weil:
  In Java sind alles Objekte (ausnahme primitive Datentypen).
  Java kann Klassen und Vererbung.
  (primitive Datentypen nicht Objektorientiert, plus ein paar andere Sachen, nur mit Reflections Klassenbeschreibungen veränderbar)

4. funktionales Paradigma [21P]
   a) 3/4 Eigenschaften fkt. Programmierung

   - Lösungsansatz:
     Kein Kontext
     Keine Seiteneffekte
     Keine Schleifen (sondern Rekursion)
     Alles ist eine Funktion

   b) Was ist ein induktiver Datentyp (leerer + rek. Konstruktor)

   - Lösungsansatz:
     Es gibt einen leeren Konstruktor (der z.B. eine Leere Liste erzeugt)
     Und einen rekursiven Konstruktor, der den leeren aufruft und dann mit Werten füllt (z.B. Elemente in die Liste einfügen)

   c) Pattern matching (5 Beispiele, disjunkt, vollständig/exhaustive & wann ist das der Fall)

   - Lösungsansatz:
     Disjunkt: jeder mögliche Wert wird von maximal einem Fall abgedeckt
     Vollständig: jeder mögliche Wert wird von mindestens einem Fall abgedeckt
     Beides: jeder mögliche Wert wird von genau einem Fall abgedeckt.

   d) Entscheiden + Begründen Skala Beispiele rekursiv, linear-rek., endrek.

   - Lösungsansatz:
     rekursiv: Die Funktion ruft sich selbst auf
     linear-rekursiv: Die Funktion ruft sich genau ein mal selbst auf
     endrekursiv: (mehrere Definitionen möglich, z.B.)
     Der selbstaufruf ist das letzte, was in der Funktion passiert.
     Beim Stack auflösen muss nur noch das Ergebnis durchgereicht werden.

   e) Business Case Setting: warum Paradigma hier gut oder aua
   f) Was sind Fkt. höherer Ordnung, Unterversorgung, Currying?

   - Lösungsansatz:
     Funktion höherer Ordnung: Eine Funktion, die eine andere Funktion zurück gibt.
     Unterversorgung: partielles „Berechnen“ einer Funktion trotz Auslassen von Parametern mittels Platzhaltersymbol (_)
     Currying: Darstellung einer Funktion mit n Parametern durch Aufrufhierarchie einstelliger Funktionen der Tiefe n

5. logisches Paradigma [15P]

a) Was bedeuten Unifizierbarkeit und Unifikation? [3P]

- Lösungsansatz:
  Unifikation: finden von Substitutionsschritten, sodass zwei prädikatenlogische Terme gleich werden
  Unifizierbar: Es gibt eine Liste von Substitutionsschritten, sodass die Terme gleich werden
  Unifikator: Die Liste dieser Substitutionsschritte

b) aus gegebener Liste Element k entfernen und Liste und Element zurück geben
removeAt(X, [1,2,3,4,5], 3, R)[5P]
c) gegebenes Programm SLD Resolution, Hornausdruck [7P wegen Schreibaufwand]
(siehe Übungsaufgaben 8.3 und 9.2)

6. paralleles Paradigma[11P]
   a) SIMD, MIMD, Flynnsches Taxonomien[4P]
   b) Abhängigkeiten zwischen Instruktionen, paarweise vergleichen [7P]
   c) ggf. Speicher statt Anderem