Übungsklausur „Programmiersprachen 1“

Beide Teile

Hinweis zum Umfang: Diese Übungsklausur bezieht sich auf gestellte Übungsaufgabe, der zu analysierende Code ist dementsprechend umfangreicher, als er in einer praktischen Klausur wäre.

Erlaubte Hilfsmittel: Taschenrechner

1. Implementierung von Funktionen

Die folgenden Übungsaufgaben von der Webseite (http://www.fh-wedel.de/mitarbeiter/mri/prog-1/ws16/) sind unmittelbar relevant für die Klausur. Sofern Sie diese bisher nicht gelöst haben, empfehle ich **dringend** dies nachzuholen.

16. Übungsaufgabe „Hangman“
17. Übungsaufgabe „Rezepte“
19. Übungsaufgabe „Beliebige Funktionen“
20. Übungsaufgabe „Code zu Syntaxdiagammen“

2. Analyse von PASCAL-Programmen 5 Punkte

Im Folgenden sehen Sie die Deklarationen verschiedener Funktionen, deren Implementierung sie gemäß den Kommentaren als korrekt voraussetzen können. Sofern für eine Funktion die Implementierung gezeigt wird, müssen Sie die Wirkung selbständig folgern.

Kreuzen Sie die **5 (in Worten: fünf)** wahren Aussagen an. Für jedes korrekt gesetzte Kreuz gibt es einen Punkt, sollten Sie mehr als fünf Kreuze machen, erhalten Sie für die Aufgabe 0 Punkte.

Hinweis: Das zu sehenden Programm lässt sich fehlerfrei kompilieren.

```
1   program klausur_eins;
2
3   type
4      TByteSet = set of byte;
5      TCapitalCharSet = set of 'A' .. 'Z';
6
7   // Diese Funktion berechnet, ob der übergebene Buchstabe eine binäre
8   // Ziffer (0 oder 1) ist.
9   function istBinaerZiffer(c : char) : boolean;
10
11  // Diese Funktion berechnet, ob der übergebene Buchstabe eine
12  // hexadezimale Ziffer ist. Dabei sollen für den Buchstabenanteil
13  // sowohl Klein- als auch Großbuchstaben für die "Ziffern" A-F
14  // zulässig sein.
15  function istHexZiffer(c : char) : boolean;
16
17  // Diese Funktion berechnet die Summe aller der in der Menge
18  // enthaltenenen Elemente.
19  function mengenSumme(menge : TByteSet) : Integer;
20
21  // Diese Funktion berechnet die Anzahl der in der Menge
22  // enthaltenenen Elemente. Doppelt in die Menge eingefügte
23  // Werte sind dabei nur einfach zu zählen.
24  function mengenAnzahl(menge : TByteSet) : byte;
25
26  function klausur1(input : string) : TCapitalCharSet
27  var
28     toReturn : TCapitalCharSet;
29     c        : char;
30  begin
31     toReturn := [];
32     for c in input do
33        toReturn := toReturn + [upcase(c)];
34
35     klausur1:= toReturn;
36  end;
37
```

- ☐ Die Funktionen `istBinaerZiffer` und `istHexZiffer` werden für identische Parameter niemals identische Resultate berechnen.
- ☐ Die Funktion `istBinaerZiffer` könnte in der Implementierung der Funktion `istHexZiffer` zum Einsatz kommen.
- ☐ `High(Integer)` ist ein praktisch mögliches Ergebnis für die Funktion `mengenSumme`.
- ☐ Der Aufruf `istHexZiffer('8')` ergibt `true`.
- ☐ Der Aufruf `istBinaerZiffer('2')` ergibt `false`.
- ☐ Der Rückgabetyp der Funktion `mengenSumme` ist unnötig groß, er könnte auch `Byte` sein.
- ☐ Die Prozedur `klausur1` gibt alle Buchstaben des übergebenen Strings aus.
- ☐ In dem zu sehenden Code werden zu keinem Zeitpunkt Funktionen aufgerufen.
- ☐ In Zeile 15 wird eine Prozedur mit einem Parameter deklariert.
- ☐ In Zeile 31 wird die lokale Variable `toReturn` mit der leeren Menge initialisiert.
- ☐ Das Ergebnis der Funktion `klausur1` ist für die Parameter „ABC“ und „cba“ identisch.
- ☐ Die Funktion `klausur1` kann mit dem Parameter „a1“ aufgerufen werden und wird ein sinnvolles Ergebnis berechnen.

3. Schreibtischtest von PASCAL-Programmen

```
program klausur_probe;

function klausur2(albert : char; berta : string) : string;
begin
   while(pos(albert + albert, berta) > 0) do
      delete(berta, pos(albert, berta), 1);

   klausur2 := berta;
end;

begin
   writeln(klausur2('a', 'Saalfeld'));
   writeln(klausur2('a', 'Salfeld'));

   writeln(klausur2('t', 'Butterpresse'));
   writeln(klausur2('t', 'Erdbeertorte'));

   writeln(klausur2('e', 'Butterpresse'));
   writeln(klausur2('e', 'Erdbeertorte'));

   writeln(klausur2('s', 'Butterpresse'));
   writeln(klausur2('s', 'Erdbeertorte'));

   writeln(klausur2('K', 'TKKG'));
   writeln(klausur2('k', 'TKKG'));
end.
```

Welche Ergebnisse wird dieses Programm ausgeben, wenn es gestartet wird?

1. _____
2. _____
3. _____
4. _____
5. _____
6. _____
7. _____
8. _____
9. _____
10. _____

4. Schreibtischtest von Pascal Programmen

```
program klausur_probe;

function klausur3(x : double) : double;
begin
   klausur3 := abs(sqrt(power(x * 3, 4)) + 4 * x + 3) / cos(x);
end;

var
   i : Integer;
begin
   for i := -2 to 2 do
      writeln(klausur3(i):5:2);
end.
```

Hinweise zu den verwendeten Funktionen:

- `abs(x)` berechnet den Betrag von x => $|x|$
- `sqrt(x)` berechnet die Qudratwurzel von x => $\sqrt{x}$
- `power(x, y)` => x^y

a) Mathematische Schreibweise

Notieren Sie die von `klausur3` implementierte Formel in der üblichen mathematischen Schreibweise.

b) Ergebnisse

Welche Ergebnisse wird dieses Programm ausgeben, wenn es gestartet wird?

1. _____
2. _____
3. _____
4. _____
5. _____

5. Implementierung von Syntax-Diagrammen als PASCAL-Programm

Eine Bankverbindung nach dem „International Bank Account Number“-Standard IBAN setzt sich in Deutschland aus vier Teilen zusammen. Bis auf den Ländercode handelt es sich bei allen Bestandteilen des Codes um Ziffern. Diese Teile werden in der folgenden Reihenfolge zusammengesetzt:

1. Der Ländercode lautet stets „DE“, „ME“ oder „RS“.
2. Die zweistellige Prüfsumme
3. Die achtstellige Bankleitzahl.
4. Die zehnstellige Kontonummer.

`DE44500105175407324931` wäre nach diesen Regeln eine gültige IBAN, `DE1337` hingegen nicht.

---

c) Zeichnen Sie ein Syntaxdiagramm

Überführen Sie obige Beschreibung in ein Syntaxdiagramm. Verwenden Sie dabei für jede unterstrichene Bezeichnung eine eigene Regel. Die Regeln Ziffer und Buchstabe können Sie als gegeben voraussetzen und gegebenenfalls verwenden. Die Verwendung von selber definierten „Hilfsregeln“ die nicht in der Aufgabenstellung erwähnt werden ist erlaubt.

d) Implementierung als PASCAL-Programm

Diese Aufgabe ist am Rechner zu lösen. Laden Sie Ihr Ergebnis danach gerne auf den Abgabeserver.

Setzen Sie **Ihr Syntaxdiagram**m als PASCAL Programm um und beachten Sie dabei die folgenden Maßgaben:

- Schreiben Sie die Nummer dieser Aufgabe, Ihren Namen und Ihre Matrikelnummer als Kommentar in den Quelltext.
- Implementieren Sie die bisher vorgegebenen Regeln „Ziffer“ und „Buchstabe“ als eigene Funktionen. Diese Nehmen einen einzelnen Buchstaben (`Char`) entgegen und berechnen daraus einen Wahrheitswert (`Boolean`).
- Jede von Ihnen definierte Regel ist als eine eigene Funktion zu implementieren denen ein `String` übergeben wird woraus dann ein `Boolean` berechnet wird. Der Name dieser Funktionen leitet sich aus dem Namen der Regel ab.
- Das Hauptprogramm muss keinerlei Benutzereingaben entgegennehmen. Stattdessen rufen Sie die Funktion Bankverbindung mit mindestens den im Aufgabentext beispielhaft genannten Werten auf und prüfen so die korrekte Implementierung. Ihre Ausgabe könnte dabei aussehen wie folgt:

  ```
  DE44500105175407324931 -> True
  DE1337 -> False
  ```

  Die Werte `True` und `False` werden dabei natürlich von Ihrer Funktion berechnet.