Lösung der Informatikprüfung Klasse 4 Sa Kantonsschule XY 2007

Name :.....Vorname : .....

Du hast 90 Minuten Zeit. Spicken ist nicht erlaubt (Die Prüfung wird sofort eingezoegen und Deine mögliche Bestnote wird sofort zu einer 4). Hilfsmittel sind die ausgeteilten Skripte dieser Woche. Die Computer bleiben ausgeschaltet. Achte auf eine saubere Darstellung. Deine vorgeschlagene Lösung muss eindeutig und klar ersichtlich sein.

Aufgabe	1	2	3	4	5	6	7	8	9	10	Total
Punkte	2	2	1	4	4	3	3	3	4	4	30

1. **Aufgabe** :

   Was heisst das, wenn eine Variable als *int* (sog. *integer*) definiert wird ? Mache Beispiele !

   Antwort :

   int definiert die Variable als eine ganze Zahl.
   Bsp. : $\{... -2, -1, 0, 1, 2, 3...\} = \mathbb{Z}$.

   Bewertung :

   $\mathbb{Z} \rightsquigarrow 0.5$ Pkte. / Bsp. $\rightsquigarrow 0.5$ Pkte.

   Was kennst Du sonst noch ausser *int* ? Mache Beispiele !

   Antwort :

   Wir haben auch *real* kennengelernt. *real* definiert die Variable als eine reelle Zahl.
   Bsp. : $\pi, 0, 3.5, -11.123456789$ usw. Mathematisch : alle $x \in \mathbb{R}$

   Bemerkung : Der Computer rundet die reellen Zahlen immer auf oder ab, da sein Speicher endlich ist ! 1/3 ist für ihn z.B. ‘genau’ 0.333333333 und endet mit der neunten Kommastelle.

   Bewertung :

   $\mathbb{R} \rightsquigarrow 0.5$ Pkte. / Bsp. $\rightsquigarrow 0.5$ Pkte.

2. **Aufgabe** :

   Was heisst auf Deutsch *random* und was heisst auf Deutsch *square root* ?

   Antworten :

   Random : Zufall, um Zufallszahlen zu generieren, z.B. bei der Modellisation des Münzenwurfes.

   Square root : Quadratwurzel, z.B. $sqrt(25) = \sqrt{25} = 5$.

   Bewertung : Pro Wort $\rightsquigarrow$ 1 Pkt.

3. Was ist das *EVA-Prinzip* ?

   Antwort :

   Das *EVA-Prinzip* (Eingabe - Verarbeitung - Ausgabe) gilt als Grundschema der elektronischen Datenverarbeitung (EDV).

   Bewertung : Pro Wort $\rightsquigarrow$ 1/3 Pkt.

4. **Aufgabe** :

   Im folgenden JavaKara-Programm hat es 4 Syntaxfehler. Welche ?

   Korrigiertes Program :

```
void invertLeaf()   {
   if (kara.onLeaf()) {
      kara.removeLeaf();
   }
   else {
      kara.put.leaf();
   }
}

public void myProgram() {
   invertLeaf();
   while (!kara.treeFront()) {
      kara.move();
```

```
        invertLeaf();
    }
}
```

Bemerkung : In der Prüfung existieren nicht nur 4, sondern 5 Syntaxfehler !

Bewertung : Pro gefundener Syntaxfehler ↬ 1 Pkt. (max. 4 Pkte.)

5. **Aufgabe** :

Erkläre, was das Programm in der vorigen Aufgabe mit dem Objekt "*kara*" macht.

Erklärung des Programmes :

Das JavaKara-Programm besteht grundsätzlich aus einem Hauptprogramm (*public void myProgramm()*) und einer Methode (*void invertLeaf()*).

Die Methode (*invertLeaf)* befiehlt dem Objekt "*kara*" ein Blatt zu legen, falls an dessen Aufenthaltsort kein Blatt ist, oder umgekehrt, ein Blatt aufzunehmen, falls sich am Aufenthaltsort ein Blatt befindet.

Im Hauptprogramm (*public void myProgramm*) wird als aller erstes die Methode *invertLeaf* aufgerufen. Dann, solange das Objekt "*kara*" keinen Baum vor sich hat, wird eine *while-Schlaufe* ausgeführt. In der Schlaufe selbst wird als erstes die Methode *void invertLeaf()* aufgerufen, wonach das Objekt "kara" um ein Feld nach vorne geschoben wird.

Alles in allem kann wie folgt zusammengefasst werden : Kara, der Käfer, geht bis zum nächsten Baum gerade aus, und er legt ein Blatt nieder, falls keines da ist - respektive - nimmt eines auf, falls eines am Boden liegt.

Bewertung :

Erkennen der Programmstruktur (Mindestens eines der Worte : Hauptprogramm, Methode, Subroutine, Unterprogramm oder Funktion muss in der Lösung erwähnt werden) ↬ 1 Pkt.

Erklären des Hauptprogrammes ↬ 1 Pkt.

Erklären der Methode ↬ 1 Pkt.

Übersichtlichkeit und Klarheit (keine Widersprüche und gutes Deutsch) ↬ 1 Pkt.

6. **Aufgabe** :

Das folgende JavaKara-Program ist das “Slalomprogramm". Stelle Dir vor, es wären nur drei Slalombäume da. Der mittlere Baum wird jetzt entfernt. Schreibe ein JavaKara-Programm, das den Käfer vom einten Baum zum anderen laufen lässt, wobei der Käfer eine Linksumrundung der Bäume macht. Gib auch an, wie und wo man den Käfer zu Beginn hinsetzen muss !

```
while(true) {
   while(kara.treeLeft() && !kara.treeRight()) {
      viertelDrehungLinks();
      }
      viertelDrehungRechts();
   while (!kara.treeLeft() && kara.treeRight()) {
      viertelDrehungRechts();
      }
      viertelDrehungLinks();
}
```

Mögliche Lösung :

```
while(true) {
   while( !kara.treeFront()) {
      kara.move;
      }
      kara.turnRight;
   for ( i=0; i<4; i++ ) {
      viertelDrehungLinks();
     }
      kara.turnRight();
}
```

Kara ist zu Beginn irgendwo zwischen den Bäumen und schaut gerade auf einen der beiden Bäume zu.

Bewertung :

Korrekte Grundidee ↬ 1 Pkt.

Es wird angegeben, wo und wie Kara zu Beginn platziert wird ↬ 1 Pkt.

Das Programm würde funktionieren ↬ 1 Pkt.

7. **Aufgabe** :

   In der Programmiersprache *Pascal* hast Du drei versch. Schleifen kennengelernt. Gib den genauen Syntax dieser drei Schleifen an.

   Lösung :

```
{forSchlaufe}
for i:=Startwert to Endwert do
   begin
      Befehl(e);
   end;

{whileSchlaufe}
while Bedingung(en) do
   begin
      Befehl(e);
   end;

{repeatSchlaufe}
repeat
      Befehle;
until Bedingung(en);
```

   Bewertung :

   Pro Schlaufe mit korrektem Syntax ↬ 1 Pkt.

8. **Aufgabe** :

   Mache kurze, genaue aber dennoch genügende Kommentare, in eigenen Worten, inwiefern sich die drei Schleifen, beschrieben in der vorhergehenden Aufgabe, unterscheiden.

Lösung :

Die For-Schleife ist eine Kontrollstruktur, mit der man eine Gruppe von Anweisungen (Block) mit einer bestimmten Anzahl von Wiederholungen ausführen kann.

Die While-Schleife, als Kontrollstruktur, dient dazu, eine Abfolge von Anweisungen mehrfach auszuführen, solange eine Bedingung erfüllt ist. Diese Bedingung wird geprüft, bevor die Anweisungsfolge abgearbeitet wird. Es kann also auch sein, dass die Abfolge gar nicht ausgeführt wird. Wenn die Bedingung ständig erfüllt ist, dann wird die Schleife zur Endlosschleife. Wenn deutsche Befehlsworte gewählt werden, dann heißt die While-Schleife meist Solange-Schleife.

Neben der gebräuchlicheren anfangsprüfenden gibt es in vielen Sprachen auch die endprüfende While-Schleife (Repeat-until). Mit dieser lässt sich die mindestens einmalige Ausführung des eingeschlossenen Code-Blocks erzwingen.

(www.wikipedia.org)

Bewertung :

For-Schleife : "...bestimmte Anzahl von Wiederholungen.." o.ä. ↬ 1 Pkt.

While-Schleife : "..solange als Bedingung erfüllt..." o.ä. ↬ 1 Pkt.

Repeat-Schleife : "...Endbedingung..." o.ä. ↬ 1 Pkt.

9. **Aufgabe** :

Schreibe in der Programmiersprache *Pascal* ein Programm das die Summe aller Zahlen von 10 bis 21 berechnet. Du vewendest dazu eine Schleife. Das Programm darf Benutzerunfreundlich sein, d.h. ein VA-Prinzip an Stelle des klassischen EVA-Prinzipes genügt.

Mögliche Lösung :

```
program MeinProgrammHeisstMacheSumme
```

```
var i,r :Integer;
r:=0;

begin
   for i := 10 to 21 do
     r:=r+i;
end.

writeln('Hier ist das Resultat: ' , r:2:2);
```

Bewertung :

Richtiger Syntax $\hookrightarrow$ 2 Pkt.

Richtige Idee $\hookrightarrow$ 1 Pkt.

Das Programm würde mit richtigem Syntax funktionieren $\hookrightarrow$ 1 Pkt.

10. **Aufgabe** :

Schreibe in Pascal ein von Dir selbst erfundenes Programm, das die **if-then-else-Struktur** beinhaltet.

Mögliche Lösung :

```
program MeinProgrammHeisstSo

var a,b :Real;

begin
writeln('Bitte geben Sie eine erste Zahl ein: ');
readln(a);

writeln('Bitte geben Sie eine zweite Zahl ein: ');
readln(b);

if a > b then
  writeln('a ist grösser als b!');
else
```

```
    writeln('b ist grösser/gleich a');

end.
```

Bewertung :

Ein falscher Syntax gibt einen Abzug von 0.5 Pkte., wobei man maximal 2 Pkt. verlieren kann. D.h. richtiger Syntax $\rightsquigarrow$ 2 Pkt.

Gute Idee $\rightsquigarrow$ 1 Pkt.

Das Programm funktioniert $\rightsquigarrow$ 1 Pkt.

Die Notengebung ist klassisch :

$$\mathrm{N} = 5\left(\frac{\text{Erreichte Punktzahl}}{\text{Maximal moegliche Punktzahl}}\right) + 1$$

☺