Programmierung WS 18
Hausaufgaben - Blatt 2

Julian Giesen (MNR 388487)
Levin Gäher (MNR 395035)
Gruppe 12

HA 2

a)

float, 0.0. 3 wird explizit und 2 implizit zu einem long konvertiert. Die Zahlen werden mit einander dividiert wobei das Ergebnis 0 ist. Das Ergebnis wird nun explizit zu einem float konvertiert.

b)

Nicht typkorrekt. x ist ein double und y ein boolean. Diese Typen sind unvergleichbar und der Ausdruck ist somit nicht typkorrekt.

c)

boolean, false. Nur ein teil des Ausdrucks true ist, der && Operator jedoch zwei true Eingaben für eine true Ausgabe benötigt.

d)

boolean, false. Der char 'a' wird implizit in den Integer 97 umgewandelt, da eine Integerdivision mit 98 durchgeführt wird. Das Ergebnis dieser Integerdivion ist 0. Der Double 4.6 wird explizit in einen Integer umgewandelt und erhält den Wert 4. Dann wird eine Floatdivision mit dem Wert 3.F (3.0) durchgeführt, wodurch die 4 nun implizit zu einem Float konvertiert wird und den Wert 4.0 annimmt. Das Ergebnis der Division ist 0.75. Der integer 0 wird implizit in einen float umgewandelt und dann mit 0.75 verglichen. Die beiden Werte sind ungleich. Somit ist das Ergebnis des Ausdrucks false.

e)

Nicht typkorrekt. Für den Vergleich ($x < z$) wird z implizit in einen double mit dem Wert 3.0 konvertiert. Der Vergleich gibt den boolean Wert false aus. Dieser Typ kann nicht mit dem Float 2.2F verglichen werden. Der Ausdruck ist somit nicht typkorrekt.

f)

Integer, 97. Der char 'a' wird explizit in Typ byte mit dem Wert 97 umgewandelt. Dieser Wert wird für einen Vergleich mit dem double x implizit in einen double mit dem Wert 97.0 konvertiert. Das Ergebnis des Vergleichs ist true. Der Bedingungsoperator gibt deswegen den char 'a' aus. result nimmt diesen Wert an, wobei der char 'a' jedoch implizit zu einem Integer mit dem Wert 97 konvertiert wird.

g)

double, 1.0 Der Bedingungsoperator gibt den ersten Wert(einen Integer mit dem Wert 1) aus. Da der zweite Wert jedoch ein double ist, wird der erste Wert implizit zu einem double mit dem Wert 1.0 konvertiert. Da der Typ von u var ist, Übernimmt u den Typ und Wert des doubles 1.0.

HA 4

Der Code ist auch im Anhang zu finden.

```
public class Main {


public static void main(String[] args) {
String mode = "";
while(!(mode.equals("Decimal") || mode.equals("Binary")))
{
mode = SimpleIO.getString("Bitte waehlen Sie aus:\n" +
"Decimal: Dezimalzahl ins Zweierkomplement konvertieren.\n" +
"Binary: 8-Bit -Zweierkomplement als Dezimalzahl darstellen.");
}
if (mode.equals("Decimal"))
{
int decimalWert = 128;
String binaryString = "";


while (!(-128 <= decimalWert && decimalWert <= 127))
{
decimalWert = SimpleIO.getInt("Bitte geben Sie eine ganze Zahl zwischen -128
}
boolean negativ = decimalWert < 0;

while(decimalWert != 0)
{
if(decimalWert % 2 == 0)
{
binaryString = "0" + binaryString;
}
else
{
binaryString = "1" + binaryString;
}
decimalWert /= 2;
}
while(binaryString.length() != 8)
{
binaryString = "0" + binaryString;
}


if(negativ)
{
binaryString = zweierKomplement(binaryString);
}
SimpleIO.output(binaryString, "Success");


}

else if (mode.equals("Binary"))
{
String binaryWert = "0";
int decimalWert = 0;
//Laengen Check
```

```
while (binaryWert.length() != 8)
{
binaryWert = SimpleIO.getString("Bitte geben Sie eine ganze Zahl im 8-Bit -Z
}

//Binaer Check
for(int i = 0; i < binaryWert.length(); i++)
{
if(binaryWert.charAt(i) != '0' && binaryWert.charAt(i) != '1')
{
SimpleIO.output("Es koennen nur Einsen und Nullen eingegeben werden.","Error
return;

}
}


boolean negativ = binaryWert.charAt(0) == '1';
if(negativ)
{
binaryWert = zweierKomplement(binaryWert);
}

int counter = binaryWert.length();
for(int i = 0; i < binaryWert.length(); i++)
{
counter--;
if(binaryWert.charAt(i) == '1')
{
decimalWert += pow(2,counter);
}

}

if(negativ)
{
decimalWert *= -1;
}

String result = "" + decimalWert;

SimpleIO.output(result,"Success");

}


}
public static int pow(int basis, int exponent)
{
if(exponent == 0)
{
return 1;
}

int temp = basis;
for(int i = 0; i < exponent - 1; i++)
{
temp *= basis;
}
```

```
return temp;
}

public static String zweierKomplement(String bString)
{
//Flippen der bits
String flippedString = "";
for(int i = 0; i < bString.length(); i++)
{
if(bString.charAt(i) == '1')
{
flippedString = flippedString + '0';
}
else
{
flippedString = flippedString + '1';
}
}

//Eins addieren
return(binaerAddieren(flippedString, "00000001"));


}
public static String binaerAddieren(String a, String b)
{

String result = "";
boolean carry = false;
for(int i = a.length() - 1; i>=0; i--)
{
int x = a.charAt(i);
int y = b.charAt(i);
if(x+y == 98)
{
if (carry)
{
result = "1" + result;
}
else
{
result = "0" + result;
carry = true;
}


}
else if(x+y == 97)
{
if (carry)
{
result = "0" +result;
}
else
{
result = "1" +result;
}
}
else if(x+y == 96)
```

```
{
if (carry)
{
result = "1" +result;
carry = false;
}
else
{
result = "0" +result;
}
}
}



return result;
}

}
```

HA 6

$< x \geq 0 >$
$< x \geq 0 \wedge -x \leq 0 \wedge -x = -x \wedge x = x >$
$res = -x;$
$< x \geq 0 \wedge res \leq 0 \wedge res = -x \wedge x = x >$
$c = x;$
$< x \geq 0 \wedge res \leq 0 \wedge res = -x \wedge c = x >$

$< res = -x + \sum_{k=x}^{c+1} 2k \wedge c \geq 0 >$
$while(c > 0)$
$< res = -x + \sum_{k=x}^{c+1} 2k \wedge c \geq 0 \wedge c > 0 >$

$< res + 2 * c = -x + \sum_{k=x}^{c-1+1} 2k \wedge c - 1 \geq 0 >$
$res = res + 2 * c;$
$< res = -x + \sum_{k=x}^{c-1+1} 2k \wedge c - 1 \geq 0 >$
$c = c - 1;$
$< res = -x + \sum_{k=x}^{c+1} 2k \wedge c \geq 0 >$
}
$< res = -x + \sum_{k=x}^{c+1} 2k \wedge c \geq 0 \wedge \neg(c > 0) >$
$< res = -x + \sum_{k=x}^{1} 2k >$

HA 8

$< \ b \geq 0 \ >$
$< \ b \geq 0 \ \wedge \ b \geq 0 \ \wedge \ b = b \ \wedge \ a = a \ \wedge \ 1 = 1 \ \wedge \ x \in \mathbb{Z} \ >$
$x = b;$
$< \ b \geq 0 \ \wedge \ x \geq 0 \ \wedge \ x = b \ \wedge \ a = a \ \wedge \ 1 = 1 \ \wedge \ x \in \mathbb{Z} \ >$
$res = a;$
$< \ b \geq 0 \ \wedge \ x \geq 0 \ \wedge \ x = b \ \wedge \ res = a \ \wedge \ 1 = 1 \ \wedge \ x \in \mathbb{Z} \ >$
$y = 1;$
$< \ b \geq 0 \ \wedge \ x \geq 0 \ \wedge \ x = b \ \wedge \ res = a \ \wedge \ y = 1 \ \wedge \ x \in \mathbb{Z} \ >$

$< \ res = a + b - x * y \ \wedge \ x \geq 0 \ \wedge \ x \in \mathbb{Z} \ >$
$while(x > 0) \ \{$
$< \ res = a + b - x * y \ \wedge \ x > 0 \ \wedge \ x \in \mathbb{Z} \ >$

$if(x\%2 == 0) \ \{$
$< \ res = a + b - x * y \ \wedge \ x > 0 \ \wedge \ x \in \mathbb{Z} \ \wedge \ x\%2 = 0 \ >$

$< \ res = a + b - \frac{x}{2} * 2y \ \wedge \ \frac{x}{2} > 0 \ \wedge \ \frac{x}{2} \in \mathbb{Z} \ >$
$y = 2 * y;$
$< \ res = a + b - \frac{x}{2} * y \ \wedge \ \frac{x}{2} > 0 \ \wedge \ \frac{x}{2} \in \mathbb{Z} \ >$
$x = x/2;$
$< \ res = a + b - x * y \ \wedge \ x > 0 \ \wedge \ x \in \mathbb{Z} \ >$
$\}$
$else \ \{$
$< \ res = a + b - x * y \ \wedge \ x > 0 \ \wedge \ x \in \mathbb{Z} \ \wedge \ x\%2 = 1 \ >$

$< \ res + y = a + b - \frac{x-1}{2} * 2y \ \wedge \ \frac{x-1}{2} \geq 0 \ \wedge \ \frac{x-1}{2} \in \mathbb{Z} \ >$
$res = res + y;$
$< \ res = a + b - \frac{x-1}{2} * 2y \ \wedge \ \frac{x-1}{2} \geq 0 \ \wedge \ \frac{x-1}{2} \in \mathbb{Z} \ >$
$y = 2 * y;$
$< \ res = a + b - \frac{x-1}{2} * y \ \wedge \ \frac{x-1}{2} \geq 0 \ \wedge \ \frac{x-1}{2} \in \mathbb{Z} \ >$
$x = (x - 1)/2;$
$< \ res = a + b - x * y \ \wedge \ x \geq 0 \ \wedge \ x \in \mathbb{Z} \ >$
$\}$

$< \ res = a + b - x * y \ \wedge \ x \geq 0 \ \wedge \ x \in \mathbb{Z} \ >$
$\}$
$< \ res = a + b - x * y \ \wedge \ x \geq 0 \ \wedge \ x \in \mathbb{Z} \ \wedge \ \neg(x > 0) \ >$
$< \ res = a + b \ >$

b)

Variante V = x
Bedingung für die einzige Schleife: $B = x > 0$
$B \Rightarrow V > 0$, denn $B \iff x > 0 \iff V > 0$

Beweis, dass jede Iteration gilt: $x \mapsto x_n \implies x_n < x$
$< \ x = x_n \wedge x > 0 \wedge x \in \mathbb{Z} \ >$
$if(x\%2 == 0)\ \{$
$Zweig\ A:$
$< \ x = x_n \wedge x > 0 \wedge x \in \mathbb{Z} \wedge x\%2 == 0 \ >$
$< \ x = x_n \wedge \frac{x}{2} > 0 \wedge \frac{x}{2} \in \mathbb{Z} \ >$
$y = 2 * y;$
$< \ x = x_n \wedge \frac{x}{2} > 0 \wedge \frac{x}{2} \in \mathbb{Z} \ >$
$x = x/2;$
$< \ x = x_n \wedge x > 0 \wedge x \in \mathbb{Z} \ >$
$\}$
$else\ \{$
$Zweig\ B:$
$< \ x = x_n \wedge x > 0 \wedge x \in \mathbb{Z} \wedge x\%2 == 1 \ >$
$< \ x = x_n \wedge \frac{x-1}{2} \geq 0 \wedge \frac{x-1}{2} \in \mathbb{Z} \ >$
$res = res + y;$
$< \ x = x_n \wedge \frac{x-1}{2} \geq 0 \wedge \frac{x-1}{2} \in \mathbb{Z} \ >$
$y = 2 * y;$
$< \ x = x_n \wedge \frac{x-1}{2} \geq 0 \wedge \frac{x-1}{2} \in \mathbb{Z} \ >$
$x = (x-1)/2;$
$< \ x = x_n \wedge x \geq 0 \wedge x \in \mathbb{Z} \ >$
$\}$

Zweig A = A(x):
$x = x_n \wedge x > 0 \wedge x \in \mathbb{Z}$
$\iff x_n > 0$
$\implies$ Zweig A kann die Schleife nicht terminieren

Zweig B = B(x):
$x = x_n \wedge x \geq 0 \wedge x \in \mathbb{Z}$
$\iff x_n \geq 0$
$\implies$ Nur Zweig B kann die Schleife terminieren

Beweis, das Zweig B unausweichlich ausgeführt wird und terminiert:
Seien $A(x)$ und $B(x)$ mathematische Funktionen, die das Verhalten von x über Zweig A bzw. Zweig B beschreiben:
$A(x) = \frac{x}{2} \ \ \forall \frac{x}{2} \in \mathbb{Z}$
$B(x) = \frac{x-1}{2} \ \ \forall \frac{x}{2} \notin \mathbb{Z}$
Dann sei $W(x)$ eine zusammengesetzte Funktion aus $A(x)$ und $B(x)$, die das Verhalten von x über eine gesamte Schleifen-Iteration beschreibt:
$W(x) = \begin{cases} A(x) & \frac{x}{2} \in \mathbb{Z} \\ B(x) & \frac{x}{2} \notin \mathbb{Z} \end{cases}$
Dann ist $W(x)^{it}$ eine rekursive Funktion zur Beschreibung von x über die gesamte Schleife bis zu einer maximalen Iteration it:
$| \{x \in \mathbb{N}_0 \mid 0 \leq x \leq b\} | \in \mathbb{N} \wedge x_n < x \implies it \in \mathbb{N} \wedge it \leq b$
$W(x)^{it}$ wiederrum lässt sich beschreiben als:
$\lim_{i \mapsto it} A(x)^i = 1$ oder bis $\frac{x}{2} \notin \mathbb{Z} \implies W(x) = B(x)$
und insbesondere $x = 1 \implies W(x) = B(1) = 0$
$\lim_{i \mapsto it} B(x)^i = 0$ oder bis $\frac{x-1}{2} \notin \mathbb{Z} \implies W(x) = A(x)$
Also muss mit $x_n < x$ und $it \in \mathbb{N} \wedge it \leq b$ folgern: $\lim_{i \mapsto it} W(x) = 0$

TL;DR:
Insbesondere gilt, dass die Anzahl der Iterationen endlich sein muss.
x kann nur eine begrenzte Anzahl an Werten annehmen:
$| \{x \in \mathbb{N}_0 \mid 0 \leq x \leq b\} | \in \mathbb{N}$
Da aber gilt: $x_n < x$ kann die Schleife nur maximal $it = b$ Iterationen durchlaufen.