

☐ Gr. 1, Dr. D. Auer☐ Gr. 2, Dr. G. Kronberger☒ Gr. 3, Dr. H. GruberName Andreas RoitherAufwand in h 6 h

Punkte _____

Kurzzeichen Tutor / Übungsleiter _____ / _____

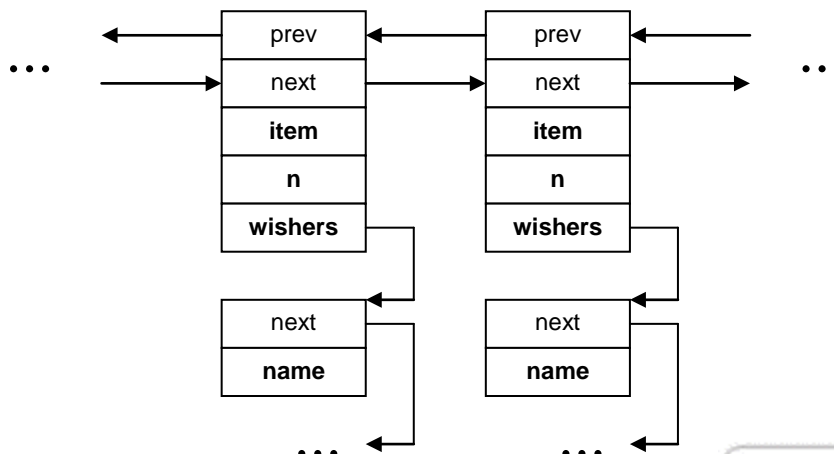
1. Christkind in *Panic Mode*: WLA, die Zweite**(12 Punkte)**

Kurz bevor das Christkind mithilfe Ihres Wunschzettelanalysators *WLA* aus der Übung 8 seinen neuen Geschäftsprozess starten will, fällt es ihm wie Sternschnuppen von den Augen: Es bastelt die Geschenke ja nicht mehr selbst und trägt sie aus, sondern bestellt alles bei Amazon und lässt von dort auch gleich zusenden. Dazu muss das Christkind bei der Bestellung aber für jeden Artikel eine Liste all jener Personen an Amazon liefern, an welche dieser gehen soll.

Zum Glück haben für Sie schon die Weihnachtsferien begonnen, so dass Sie dem Christkind noch einmal unter die Arme greifen können: Zuerst wird die Wünschedatei *Wishes.txt* so vereinfacht, dass vor den Wünschen eines Wunschzettels eine eigene Zeile mit dem Namen der Person (gefolgt von einem Doppelpunkt) steht, die diese Wunsch geäußert hat. Z. B.:

```
...
Barbara:
Barbie-Puppe
Puppenküche
Blockflöte
Christoph:
Schlitten
Matchbox-Auto
...
```

Dann ändern Sie Ihren Wunschlistenanalysator so ab, dass in jedem Wunschknoten (nun Element einer doppelt-verketteten Liste) nicht nur der Wunsch (*item*) und seine Häufigkeit (*n*) vorkommt, sondern darin auch eine einfach-verkettete Liste mit den Namen (*name*) aller „WünscherInnen“ dieses Wunsches (*wishers*) verankert ist. Dadurch ergibt sich eine doppelt-verkettete Liste, bei der in jedem Knoten eine einfach-verkettete Liste mit mindestens einem Knoten ankert, gemäß folgender Abbildung:

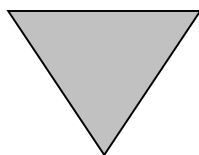


Nun kann sich das Christkind an Amazon wenden (also bestellen) und sich dann fröhlich dem Feiern widmen.

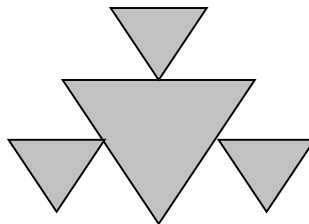


2. Ein Lichtlein brennt, ... dann vier, dann ...**(4 + 2 Punkte)**

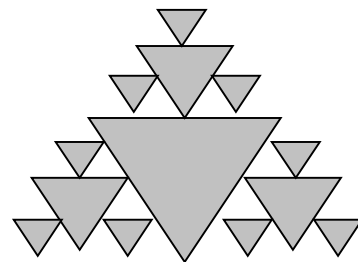
Die Anzahl der Kerzen (*candles*), die man auf einem Christbaum unterbringen kann, hängt im Wesentlichen von der Höhe (h) des Baumes ab. Studieren Sie folgende Beispiele von Christbaum-Beleuchtungen mittels Kerzen:



$h = 1$
candles = 1



$h = 2$
candles = 4



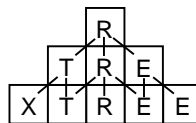
$h = 3$
candles = 13

- Geben Sie eine rekursive Definition und einen rekursiven Algorithmus für $Candles(h)$ an.
- Geben Sie eine iterative Implementierung für $Candles(h)$ an.

3. ... steht die Feuerwehr vor der Tür**(6 Punkte)**

Nicht nur das Wort *XTREE*¹, alle Worte mit einer ungeraden Anzahl von Buchstaben eignen sich für die Weihnachtsbaumrekursion: Die einzelnen Buchstaben des Wortes werden in Form eines Weihnachtsbaums angeordnet.

Das ergibt für *XTREE* z.B. folgenden Weihnachtsbaum:



Nun stellt sich ein Sicherheitsbeauftragter der Feuerwehr die Frage, wie viele mögliche Wege ein Brand bis zur Spitze des Weihnachtsbaums nehmen kann, wenn nur ein Buchstabe in der untersten Ebene durch eine Kerze entzündet wird und sich das Feuer nur nach oben oder rechts-oben (in der linken Hälfte des Baumes) bzw. links-oben (in der rechten Hälfte) ausbreiten kann.

Beantworten Sie diese Frage mit einer rekursiven mathematischen Definition der Funktion $XFire$ und implementieren Sie diese Funktion in Pascal.

Können Sie auch eine iterative Lösung (mathematische Definition und Funktion) angeben?

¹ Aus dem engl. *Xmas* abgeleiteter Ausdruck im Informatik-Kauderwelsch (engl. *compu slang*).

Übung 9

Aufgabe 1

Lösungsidee

Bei dieser Aufgabe wird eine doppelt verkettete Liste, die eine einfach verkettete Liste enthält, initialisiert. Die Liste wird am Anfang auf Nil gesetzt. Alles weitere wird in der appendToAmazonList behandelt. Falls die Liste Nil ist, wird ein neues Element für diese Liste erstellt. Bei jedem einlesen einer Text Zeile wird überprüft ob die Zeile den Namen oder den Wunsch eines Kindes enthält. Der Name und der Wunsch wird dem appendToAmazonList übergeben. Falls ein Wunsch bereits enthalten ist, wird der Name zur Wishers Liste hinzugefügt. Andernfalls wird ein neues Element erstellt mit dem Namen des Kindes als Wisher.

```

1  (* WLA:                                     HDO, 2016-12-06
   _____
3   Wish list analyzer for the Christkind.
   =====*)
5 PROGRAM WLA;
7 (*$IFDEF WINDOWS*)
   USES
9   WinCrt;
   (*$ENDIF*)
11
12 TYPE
13   wishPtr = ^wishersNode;
   wishersNode = RECORD
15     next : wishPtr;
     name : STRING;
17 END; (* END RECORD *)
19
   amazonListPtr = ^listElement;
   listElement = RECORD
21     prev : amazonListPtr;
     next : amazonListPtr;
23     item : STRING;
     n : INTEGER;
25     wishers : wishPtr;
   END; (* END RECORD *)
27
   amazonList = RECORD
29     first : amazonListPtr;
     last : amazonListPtr;
31 END; (* END RECORD *)
33
34 PROCEDURE InitAmazonList(VAR l : amazonList);
   BEGIN
35   l.first := NIL;
     l.last := NIL;
37 END;
39
40 FUNCTION newWishNode(name : STRING) : wishPtr;
   VAR temp : wishPtr;
41 BEGIN
   New(temp);

```

```

43  temp^.next := NIL;
    temp^.name := name;
45
    newWishNode := temp;
47 END;

49 PROCEDURE appendToWishList(VAR list : wishPtr; node : wishPtr);
VAR temp : wishPtr;
51 BEGIN
    IF (list <> NIL) THEN
53 BEGIN
        temp := list;
55 WHILE (temp^.next <> NIL) DO
            temp := temp^.next;
57
        temp^.next := node;
59 END
    ELSE
61 list := node;
END;

63 FUNCTION newAmazonListNode(item, wisher : STRING) : amazonListPtr;
65 VAR temp : amazonListPtr;
BEGIN
67 New(temp);
    temp^.prev := NIL;
69 temp^.next := NIL;
    temp^.item := item;
71 temp^.n := 1;
    temp^.wishers := newWishNode(wisher);
73
    newAmazonListNode := temp;
75 END;

77 (* Append to the amazon list *)
78 (* if there is no toy in the list a new element will be made *)
79 (* if a toy is already in the list, the name of the wisher will be added to the
    wisher list *)
PROCEDURE appendToAmazonList(VAR amazon_List : amazonList; toy, wisher : STRING
    );
81 VAR aListPtr, temp : amazonListPtr;
BEGIN
83 aListPtr := amazon_List.first;

85 IF amazon_List.first = NIL THEN
    BEGIN
87 aListPtr := newAmazonListNode(toy, wisher);
        amazon_List.first := aListPtr;
89 amazon_List.last := aListPtr;
    END
    ELSE
91 BEGIN
93 WHILE aListPtr^.next <> NIL DO
        BEGIN
95 IF aListPtr^.item = toy THEN break
            ELSE
97 aListPtr := aListPtr^.next;
        END;

```

```

99      IF aListPtr^.item = toy then
101      BEGIN
          appendToWishList(aListPtr^.wishers,newWishNode(wisher));
103      aListPtr^.n := aListPtr^.n + 1;
      END
105      ELSE
      BEGIN
107          aListPtr := newAmazonListNode(toy, wisher);
          aListPtr^.prev := amazon_List.last;
109          amazon_List.last^.next := aListPtr;
          amazon_List.last := aListPtr;
111      END;
      END;
113 END;

115 (*#####*)
116 (*  Printing to Console  *)
117 (*#####*)

119 PROCEDURE printWishers(wishers : wishPtr);
      VAR temp : wishPtr;
121 BEGIN
          temp := wishers;
123
          WHILE temp <> NIL DO
125              BEGIN
                  Write(temp^.name, ' ');
127                  temp := temp^.next;
              END;
129 END;

131 PROCEDURE printAmazonList(list : amazonList);
      VAR
133          temp : amazonListPtr;
          count : INTEGER;
135 BEGIN
          temp := list.first;
          count := 1;
137
          WHILE temp <> NIL DO
          BEGIN
141              WriteLn(count, '. Item:');
              WriteLn('Name > ',temp^.item, ' | Count > ', temp^.n);
143              WriteLn('Wishers: ');
              printWishers(temp^.wishers);
              WriteLn;
              WriteLn;
147              count := count + 1;
              temp := temp^.next;
149          END;
          END;
151
      VAR
153          wishesFile: TEXT;
          line, wisher, toy: STRING;
155          position : INTEGER;
          amazon_List : amazonList;

```

```

157 BEGIN (*WLA*)
159   InitAmazonList(amazon_List);

161   WriteLn(chr(205),chr(205),chr(185), ' Amazon wish list for XMas ',chr(204),chr
      (205),chr(205));
   WriteLn;

163   (* Read every line from txt *)
165   Assign(wishesFile, 'Wishes.txt');
   Reset(wishesFile);

167   REPEAT
169     ReadLn(wishesFile, line);
     position := pos(':',line);

171     IF position <> 0 THEN
173       wisher := Copy(line,1,position-1)
     ELSE
175       appendToAmazonList(amazon_List,line,wisher);

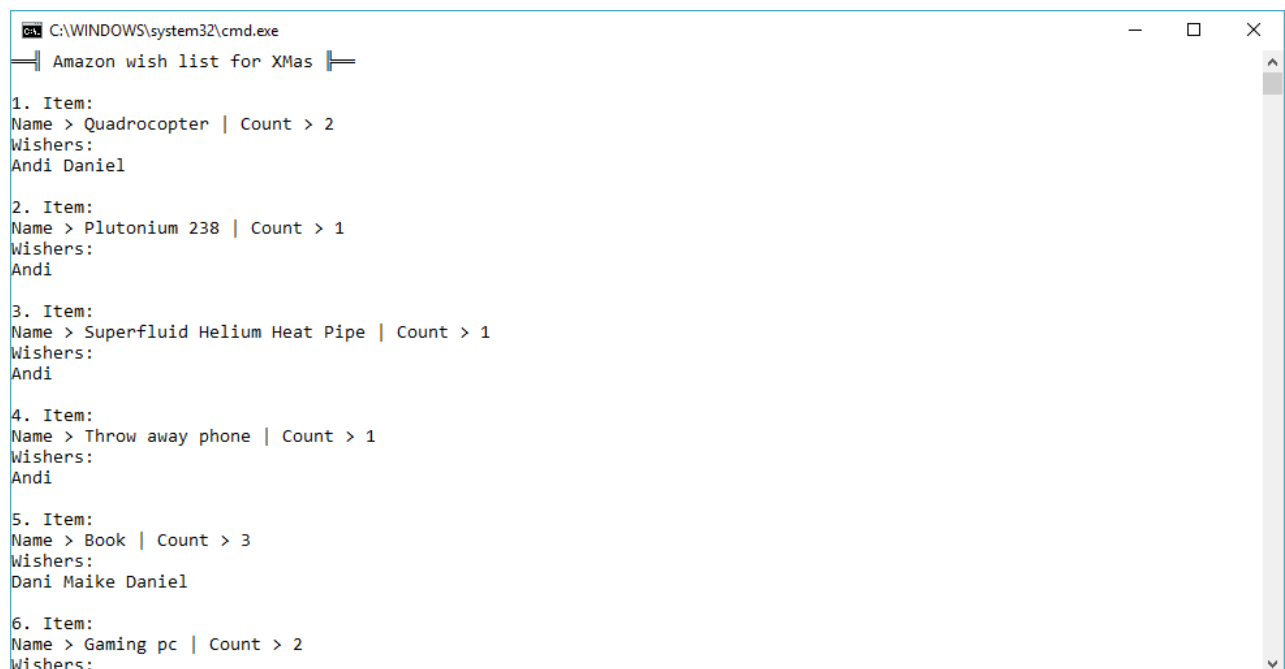
177       //WriteLn(wisher, ' ',position);
   UNTIL Eof(wishesFile);
179   Close(wishesFile);

181   printAmazonList(amazon_List);

183 END. (*WLA*)

```

XmasWish.pas



```

C:\WINDOWS\system32\cmd.exe
Amazon wish list for XMas

1. Item:
Name > Quadrocopter | Count > 2
Wishers:
Andi Daniel

2. Item:
Name > Plutonium 238 | Count > 1
Wishers:
Andi

3. Item:
Name > Superfluid Helium Heat Pipe | Count > 1
Wishers:
Andi

4. Item:
Name > Throw away phone | Count > 1
Wishers:
Andi

5. Item:
Name > Book | Count > 3
Wishers:
Dani Maike Daniel

6. Item:
Name > Gaming pc | Count > 2
Wishers:

```

Abbildung 1: Testfälle XmasWish 1



```

C:\WINDOWS\system32\cmd.exe
6. Item:
Name > Gaming pc | Count > 2
Wishers:
Dani Maike

7. Item:
Name > Nitrogen | Count > 1
Wishers:
Dani

8. Item:
Name > Learn Pascal in Three Days | Count > 2
Wishers:
Dani Maike

9. Item:
Name > Prada bag | Count > 1
Wishers:
Maike

C:\Users\Andreas\Documents\GitHub\SE-Hagenberg\1. Semester\ADE\Uebung09>

```

Abbildung 2: Testfälle XmasWish 2

```

1  Andi:
   Quadrocopter
3  Plutonium 238
   Superfluid Helium Heat Pipe
5  Throw away phone
   Dani:
7  Book
   Gaming pc
9  Nitrogen
   Learn Pascal in Three Days
11 Maike:
   Gaming pc
13 Book
   Prada bag
15 Learn Pascal in Three Days
   Daniel:
17 Book
   Quadrocopter

```

Wishes.txt

Testfälle

Eine Liste wird erstellt und ausgegeben. Die Liste wird mithilfe der Wishes.txt erstellt. Zu sehen ist ein Element der Liste und die Namen der Kinder die sich dasselbe wünschen.

Aufgabe 2

Lösungsidee

Für diese Aufgabe wird sowohl eine Rekursive sowie eine Iterative Funktion erstellt. Bei einem Rekursiven Aufruf wird die Funktion immer wieder aufgerufen bis ein Punkt erreicht bzw. eine Bedingung erfüllt ist an der der erneute Funktionsaufruf nicht mehr ausgeführt wird. Für die Berechnung wird durch beobachten der Beispiele eine mathematische Funktion erstellt. Diese lautet: 3^{h-1} wobei h die Höhe darstellt.

```

PROGRAM CandleonTree;
2  USES Math;

4  (* recursive *)
FUNCTION Candles(h: INTEGER) : INTEGER;
6  BEGIN
    IF h = 1 THEN Candles := 1 Else Candles := 3**(h-1) + Candles(h-1);
8  END;

10 (* iterative *)
FUNCTION Candles_Iterative(h : INTEGER) : INTEGER;
12 VAR candles : INTEGER;
    BEGIN
14     candles := 0;

16     WHILE h > 1 DO
        BEGIN
18         h := h - 1;
            candles := candles + 3**(h);
20     END;

22     Candles_Iterative := candles + 1;

24 END;

26 VAR result, result_it : INTEGER;
    BEGIN
28     WriteLn(chr(205),chr(205),chr(185), ' Candles on XMas Tree ',chr(204),chr(205)
        ,chr(205));

30     result := Candles(1);
        result_it := Candles_Iterative(1);
32     WriteLn('Candles with height 1: ',result, ' Iterative: ', result_it);

34     result := Candles(2);
        result_it := Candles_Iterative(2);
36     WriteLn('Candles with height 2: ',result, ' Iterative: ', result_it);

38     result := Candles(3);
        result_it := Candles_Iterative(3);
40     WriteLn('Candles with height 3: ',result, ' Iterative: ', result_it);

42     result := Candles(4);
        result_it := Candles_Iterative(4);
44     WriteLn('Candles with height 4: ',result, ' Iterative: ', result_it);

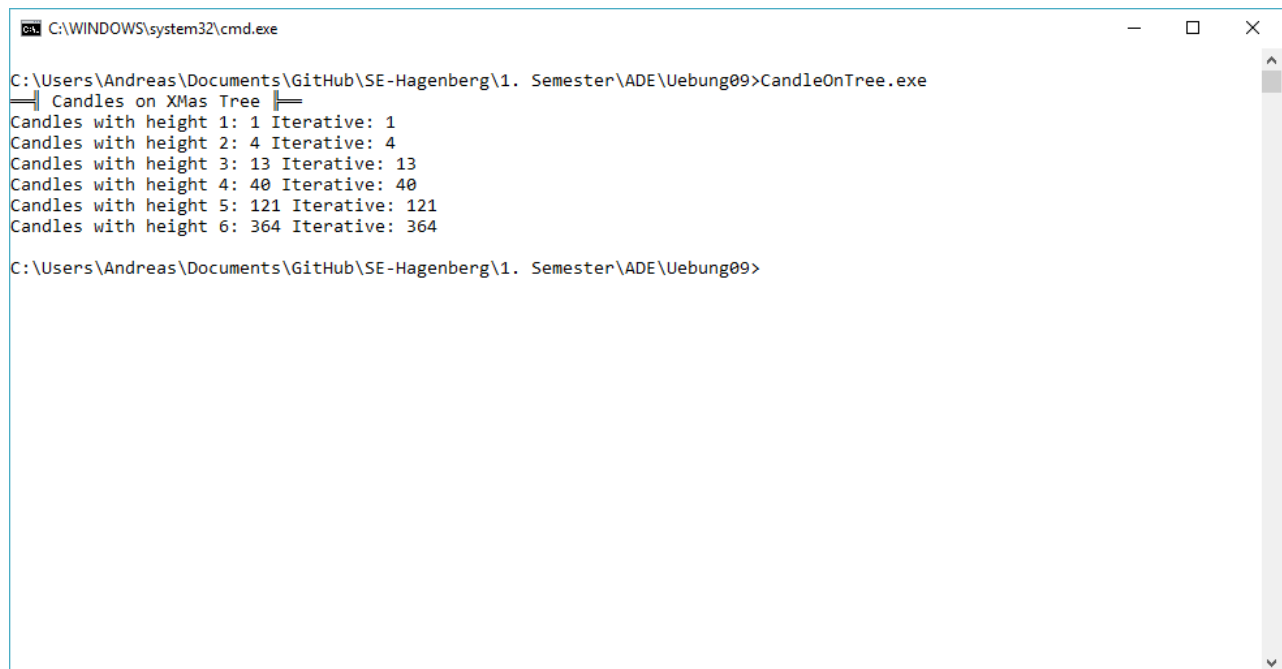
46     result := Candles(5);

```



```
48  result_it := Candles_Iterative(5);  
    WriteLn('Candles with height 5: ',result, ' Iterative: ', result_it);  
50  result := Candles(6);  
    result_it := Candles_Iterative(6);  
52  WriteLn('Candles with height 6: ',result, ' Iterative: ', result_it);  
END.
```

CandleonTree.pas



```
C:\WINDOWS\system32\cmd.exe  
C:\Users\Andreas\Documents\GitHub\SE-Hagenberg\1. Semester\ADE\Uebung09>CandleOnTree.exe  
┌─ Candles on XMas Tree ─┐  
Candles with height 1: 1 Iterative: 1  
Candles with height 2: 4 Iterative: 4  
Candles with height 3: 13 Iterative: 13  
Candles with height 4: 40 Iterative: 40  
Candles with height 5: 121 Iterative: 121  
Candles with height 6: 364 Iterative: 364  
C:\Users\Andreas\Documents\GitHub\SE-Hagenberg\1. Semester\ADE\Uebung09>
```

Abbildung 3: Testfälle CandleonTree

Testfälle

Die Testfälle zeigen die Kerzenanzahl bei entsprechender Höhe des Baumes mit der iterativen und der rekursiven Funktion.

Aufgabe 3

Lösungsidee

Bei dieser Aufgabe muss wie bei der vorherigen Aufgabe die mathematische Funktion ermittelt werden. Zusätzlich wird auch überprüft ob das eingegebene Wort ungerade und nicht 0 ist. Da bei jedem Wort das länger als 3 ist drei Wege gibt bei denen es keine Abzweigungen gibt, wird der Zähler um 3 erhöht. Danach wird der counter entsprechend mit $(\text{Länge Worte} - 3) * 2$ erhöht. Das wird solange durchgeführt bis die Länge ≤ 3 und die Abbruchbedingung erfüllt ist.

```

1 PROGRAM XmasFire;

3 (* Check if the number is uneven*)
FUNCTION CheckForOdd(i : INTEGER) : BOOLEAN;
5 BEGIN
    IF (i MOD 2 <> 0) AND (i <> 0) THEN CheckForOdd := True ELSE CheckForOdd :=
        False;
7
8 END;
9
10 (* Recursive *)
11 FUNCTION XFire(len : Integer) : INTEGER;
    VAR count : INTEGER;
12 BEGIN
13     count := 0;
14
15     IF CheckForOdd(len) THEN
16     BEGIN
17
18         IF len < 3 THEN
19             count := count + 3
20         ELSE
21             BEGIN
22                 count := count + (len - 3) * 2 ;
23                 count := count + XFire(len-2);
24             END;
25         END;
26
27         XFire := count;
28     END
29     ELSE
30     BEGIN
31         Write('Wrong Input!');
32         XFire := 0;
33     END;
34 END;
35
36 (* Iterative *)
37 FUNCTION XFire_iterative(len : INTEGER) : INTEGER;
    VAR count, temp: INTEGER;
38 BEGIN
39
40     IF CheckForOdd(len) THEN
41     BEGIN
42         count := 0;
43         temp := len;
44
45         WHILE temp > 1 DO

```

```
47     BEGIN
        count := count + 1;
49     temp := temp - 2;
    END;
51
    XFire_iterative := 3 + ((len-3) * count);
53 END
    ELSE
55 BEGIN
        Write('Wrong Input!');
57     XFire_iterative := 0;
    END;
59 END;

61 VAR word : String;
    BEGIN
63     WriteLn(chr(205),chr(205),chr(185),' Fire on XMas Tree ',chr(204),chr(205),
        chr(205));

65     word := '1234';
        WriteLn('Ways for ',word,': ',XFire(length(word)));
67     WriteLn('Ways for ',word,' Iterative: ',XFire_iterative(length(word)));

69     word := '123';
        WriteLn('Ways for ',word,': ',XFire(length(word)));
71     WriteLn('Ways for ',word,' Iterative: ',XFire_iterative(length(word)));

73     word := '12345';
        WriteLn('Ways for ',word,': ',XFire(length(word)));
75     WriteLn('Ways for ',word,' Iterative: ',XFire_iterative(length(word)));

77     word := '1234567';
        WriteLn('Ways for ',word,': ',XFire(length(word)));
79     WriteLn('Ways for ',word,' Iterative: ',XFire_iterative(length(word)));

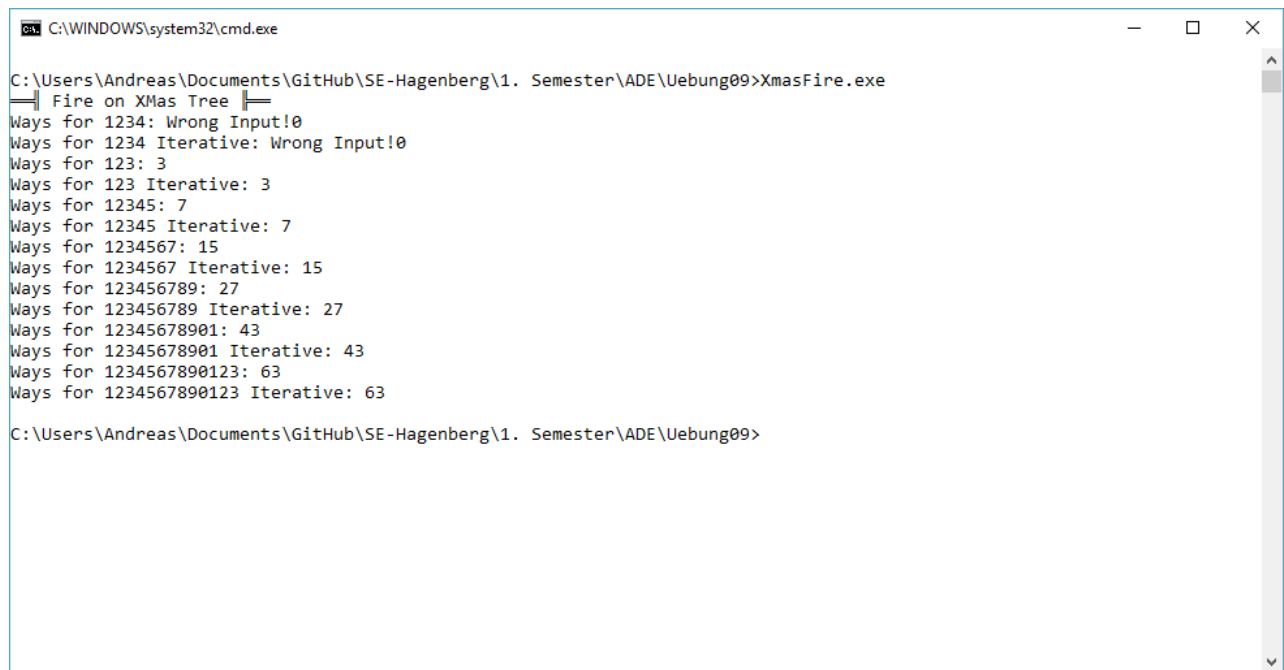
81     word := '123456789';
        WriteLn('Ways for ',word,': ',XFire(length(word)));
83     WriteLn('Ways for ',word,' Iterative: ',XFire_iterative(length(word)));

85     word := '12345678901';
        WriteLn('Ways for ',word,': ',XFire(length(word)));
87     WriteLn('Ways for ',word,' Iterative: ',XFire_iterative(length(word)));

89     word := '1234567890123';
        WriteLn('Ways for ',word,': ',XFire(length(word)));
91     WriteLn('Ways for ',word,' Iterative: ',XFire_iterative(length(word)));

93 END.
```

XmasFire.pas



```
C:\WINDOWS\system32\cmd.exe

C:\Users\Andreas\Documents\GitHub\SE-Hagenberg\1. Semester\ADE\Uebung09>XmasFire.exe
== Fire on XMas Tree ==
Ways for 1234: Wrong Input!0
Ways for 1234 Iterative: Wrong Input!0
Ways for 123: 3
Ways for 123 Iterative: 3
Ways for 12345: 7
Ways for 12345 Iterative: 7
Ways for 1234567: 15
Ways for 1234567 Iterative: 15
Ways for 123456789: 27
Ways for 123456789 Iterative: 27
Ways for 12345678901: 43
Ways for 12345678901 Iterative: 43
Ways for 1234567890123: 63
Ways for 1234567890123 Iterative: 63

C:\Users\Andreas\Documents\GitHub\SE-Hagenberg\1. Semester\ADE\Uebung09>
```

Abbildung 4: Testfälle XmasFire

Testfälle

Die Testfälle zeigen Anzahl der möglichen Wege für das jeweilige Wort.