# Antistring

Schreiben Sie eine Funktion:

**Parameter**: beliebiger String bestehend aus Buchstaben (groß/klein) und Zahlen.

**Verarbeitung**:

Antistring = alle Buchstaben wechseln die Großschreibung,

alle Ziffern werden ersetzt mit der Differenz zu 9 (0=>9,1=>8,…,9=>0),

alle Buchstaben werden ersetzt mit ihrem Gegenstück im Alphabet

(a=>z,b=>y,c=>z,…,z=>a / A=>Z,B=>Y,C=>Z,…,Z=>A)

der String wird umgedreht (abcd=>dcba).

**Rückgabe**: Der Antistring.

**Beispiel**: „abcWXY123“ => „678bcdXYZ“

# Akronym Generator

Schreiben Sie eine Funktion:

**Parameter**: Name einer Person

**Verarbeitung**:

Akronym = Standard: Akronym ist der erste Buchstaben des Vornamens

und der 1. Buchstabe des Nachnamens (beide groß).

Ausnahmen:

1. Die Mitarbeiter unserer Firma haben max. 2 Vornamen und max. 2 Nachnamen. Dann besteht das Akronym entsprechend aus 3 oder 4 Zeichen.
2. Die 2 Vornamen und die 2 Nachnamen können auch Doppelnamen sein und werden dann durch einen Bindestrich getrennt.
3. Zusatz: Im Falle des überflüssigen Adels kommt das Prefix „von“ vor. Dieses Prefix wird in das Akronym übernommen, allerdings als Kleinbuchstabe „v“.

**Rückgabe**: Akronym

**Beispiele**: „Thomas Meier“ => „TM“, „martin schmidt“ => “MS“,

„Jan-Erich Schmidt“ => „JES“, „Jan Erich Mueller“ => „JEM“

„Paul Meyer-Schmidt“ => „PMS“

„Paul von Lahnstein“ => „PvL“, „Martin von Lahnstein-Meyer“ => „MvLM“

# Dezimal zu Hexadezimal

Schreiben Sie eine Funktion:

**Parameter**: Eine Dezimalzahl.

**Verarbeitung**:

Hexadezimaldarstellung: Berechnen Sie die Hexadezimaldarstellung per Hand.

Als Erkennungsmerkmal bekommt die Hex-Zahl das Prefix

„0x“ vorangestellt.

**Rückgabe**: Hexadezimaldarstellung der Dezimalzahl.

**Beispiele**: „10“ => „0xA“, „16“ => „0x10“, „51966“ => „0xCAFE“,

„47806“ => „0xBABE“, „45054“ => „0xAFFE“

# Alternierende Strings

Schreiben Sie eine Funktion:

**Parameter**: 2 Strings mit Len > 0.

**Verarbeitung**:

Alternierender String = Gleiche Länge:

Die Zeichen des alternierenden Strings ergeben sich aus den übergebenen Strings, mit abwechselndem Bezug. Das erste Zeichen kommt aus dem 1. Parameter. (Siehe Beispiel 1)

Unterschiedliche Länge:

Die Buchstaben des kürzeren Strings werden so oft wiederholt, bis der längere String keine Zeichen mehr hat. Der alternierende String beginnt mit dem ersten Zeichen des längeren Strings und endet mit einem Zeichen des kürzeren Strings.

Einschränkung:

Leerstellen werden ignoriert. Auch bei der Berechnung der Stringlänge.

**Rückgabe**: Alternierender String aus den beiden Parametern.

**Beispiele**: „Hello“, „World“ => „HWeolrllod“

„Beer“, „Drinking“ => „DBreienrkBienegr“

„abcd“, „e f g“ => „aebfcgde“

„Biking“, „the city“ => „tBhiekciintgyB“

# Bulls and Cows

Schreiben Sie ein Programm:

**Ablauf**:

1. Das Programm generiert eine vierstellige Geheimzahl.
2. Der Spieler (Nutzer) wird nach der Geheimzahl gefragt.
3. Der Spieler gibt eine vierstellige Zahl ein (falsche Eingaben werden verworfen).
4. Das Programm prüft für jede Ziffer ob sie in der Geheimzahl vorkommt und ob sie an der richtigen Stelle steht.
5. Das Programm gibt die Anzahl der richtigen Stellen (X) und die Anzahl der vorhandenen Zahlen (Y) wie folgt aus: „X bull(s) and Y cow(s)“
6. Der Spieler hat 8 Versuche um die Zahl zu erraten.
7. Sollte der Spieler die Zahl erraten wird „You win!“ statt „4 bulls and 0 cows“ ausgegeben.
8. Sollte der Spieler alle 8 Versuche verbraucht haben, dann wird „Sorry, you’re out of turns!“ ausgegeben.

**Beispiel**: Geheimzahl: 1234 Geheimzahl: 9876

Eingabe: 1357 Eingabe: 1234

Ausgabe: 1 bull and 1 cow Ausgabe: 0 bulls and 0 cows

Eingabe: 3456 Eingabe: 5678

Ausgabe: 0 bulls and 2 cows Ausgabe: 1 bulls and 2 cows

Eingabe: 4321 Eingabe: 5690

Ausgabe: 0 bulls and 4 cows Ausgabe: 0 bulls and 2 cows

Eingabe: 3214 Eingabe: 6780

Ausgabe: 1 bull and 3 cows Ausgabe: 0 bulls and 3 cows

Eingabe: 2134 Eingabe: 8769

Ausgabe: 2 bulls and 2 cows Ausgabe: 0 bulls and 4 cows

Eingabe: 1234 Eingabe: 7869

Ausgabe: You win! Ausgabe: 1 bull and 3 cows

Eingabe: 9786

Ausgabe: 2 bull and 2 cows

Eingabe: 9867

Ausgabe: 2 bulls and 2 cows

Ausgabe: Sorry, you’re out of turns!

# Quicksum

Schreiben Sie eine Funktion:

**Parameter**: ein String (Großbuchstaben und Leerzeichen, beginnt/endet mit Großbuchstaben)

**Verarbeitung**:

Quicksum = Summe der Produkte des Zeichenwerts mit der Position,

falls andere Zeichen im String enthalten sind => 0

Zeichenwert = A => 1, B => 2, C => 3, …, Z => 26, Leerstelle => 0

**Rückgabe**: Quicksum

**Beispiele**: „ABC“ == 1\*1 + 2\*2 + 3\*3 => 14, „A B C“ == 1\*1 + 3\*2 + 5\*3 => 22

# Dotted Strings

Schreiben Sie eine Funktion:

**Parameter**: ein String (Groß-/Kleinbuchstaben)

**Verarbeitung**: dottedString = 1. lösche alle Vokale (A,E,I,O,U),

2. füge vor jedem Konsonant einen Punkt ein,

3. ersetze alle Großbuchstaben mit Kleinbuchstaben

**Rückgabe**: dottedString

**Beispiele**: „tour“ => „.t.r“, „Anwendungsprogrammierung“ => „.n.w.n.d.n.g.s.p.r.g.r.m.m.r.n.g“

# Äthiopisches Multiplizieren

Die Äthiopier hatten einen interessanten Weg gefunden, zwei große Zahlen miteinander zu multiplizieren.

Zunächst haben sie beiden Zahlen, die es zu multiplizieren galt, nebeneinander geschrieben, z.B.:

17 x 34 🡪 17 34

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Nun geht es aber etwas anderes weiter, als wir das zu tun pflegen. Die linke Zahl wird nun für jede darunterliegende Zeile halbiert, Nachkommastellen werden abgeschnitten.  Die Rechte zahl hingegen wird Zeile für Zeile verdoppelt. Dies wird so lange betrieben, bis die linke Zahl gleich 1 ist.   |  |  | | --- | --- | | 17 | 34 | | 8 | 68 | | 4 | 136 | | 2 | 272 | | 1 | 544 | | Nun werden all die Zeilen durchgestrichen, bei denen die linke Zahl gerade (also ohne Rest durch 2 teilbar) ist.   |  |  | | --- | --- | | 17 | 34 | | **~~8~~** | **~~68~~** | | **~~4~~** | **~~136~~** | | **~~2~~** | **~~272~~** | | 1 | 544 |   Von den übrig bleibenden Zeilen werden nun die rechts übrig bleibenden Zahlen miteinander addiert 🡪 34 + 544 = 578 |

Die Summe beider Zahlen ist das Ergebnis des Produkts aus 17 x 34.

**Aufgabe:** Schreiben Sie eine Funktion welche dieses Multiplizieren mit beliebigen Anfangswerten berechnet und lassen Sie dieses Ergebnis ausgeben.

# Zombie Apokalypse

Schreiben Sie eine Funktion:

Geschichte: Sie befinden sich in einer ausweglosen Situation. Auf der Flucht vor Zombies haben Sie sich am Bein verletzt und können nicht mehr laufen. N Zombies haben es auf Ihr Gehirn abgesehen. Zum Glück sind Sie ein ausgezeichneter Schütze und die Zombies ziemlich langsam. Die Zombies stehen noch X Meter von Ihnen entfernt. Sie legen pro Sekunde 0,5m zurück. Sie können pro Sekunde einen Zombie erschießen, verfügen aber nur über Y Schuss Munition.

**Parameter**: N = Anzahl der Zombies, X = Abstand in Metern, Y = Schuss Munition

Ablauf: Jede Runde erschießen Sie einen Zombie und im Anschluss bewegen sich die Zombies einen halben Meter auf Sie zu, bis:

1. Alle Zombies tot sind oder
2. die Zombies auf 0 Meter ran gekommen sind oder
3. die Munition alle ist (und noch Zombies leben).

Rückgabe: Falls Sie alle Zombies beseitigt haben: „Du hast alle N Zombies erledigt.“,  
falls die Zombies Sie erreichen bevor Sie alle erwischt haben: „Du hast M Zombies erwischt, bevor Du gefressen wurdest: Überwältigt.“,  
falls Ihnen die Munition ausgegangen ist: „Du hast M Zombies erwischt, bevor Du gefressen wurdest: keine Munition mehr.“,  
falls Sie direkt nach der letzten Kugel gefressen werden: „Du hast M Zombies erwischt, bevor Du gefressen wurdest: Überwältigt.“.

# Das Josephus Problem

Es stehen n Personen in einem Kreis. Die Personen sind nummeriert von 1 bis n. Beginnend bei Person Nummer p wird nun jede p-te Person aus dem Kreis entfernt und der Kreis danach sofort wieder geschlossen (jede Person behält dabei ihre anfänglich zugewiesene Nummer).

Parameter: n = Anzahl Personen, p = erste zu entfernende Person

Aufgabe: Geben Sie die Nummern der entfernten Personen in der Reihenfolge an, in der sie entfernt wurden, sowie zuletzt die Person die übrigbleibt.

# Arrays und noch mehr Arrays

Ihr Programm soll zwei (20-stellige) int-Arrays mit Werten befüllen.

Das erste Array soll mit Werten ab 5 in 5er-Schritten befüllt werden (5,10, 15, ...).

Das 2.Array soll die Werte des 1.Arrays \* 2 enthalten (10,20, 30, ...).

Die Befüllung ist mit einer for-Schleife auszuführen! Das Programm soll die Summen und Mittelwerte (Summe/Anzahl der Elemente) ausgeben.

Mögliche Ausgabe:

array1: Summe: ........... Mittelwert: ...........

array2: Summe: ........... Mittelwert: ...........

# Lottozahlen

Schreiben Sie ein Programm, dass 6 zufällig generierte Lottozahlen (1 bis 49) ausgibt. Bedenken Sie dabei, jede Zahl darf nur einmal vorkommen!

# Plattenleger Placido

Plattenleger Placido muss oft rechteckige Wände mit rechteckigen Platten verlegen.

Er hat seine eigene Methode entwickelt, damit er vorab jeweils nicht zu viele, aber auch nicht zu wenige Platten einkauft:

Zunächst misst er die Breite der zu verlegenden Wand und teilt diese durch die Breite der Platten. Dies ergibt für ihn die Anzahl der zu verlegenden Spalten. Natürlich rundet er dabei auf die nächste ganze Platte auf, damit die Wand komplett belegt wird. Dasselbe macht er mit der Höhe der Wand, indem er diese durch die Höhe der Platten teilt. Auch hier rundet er auf und erhält somit die zu verlegenden Reihen. Nun multipliziert er die Anzahl Spalten mit derjenigen der Reihen, was ihm die effektive Anzahl einzukaufender Platten ergibt.

Am Schluss fügt Placido dem Resultat noch 5% Reserve hinzu. Die Reserve ist wichtig, denn einige Platten gehen beim Transport, andere beim Verlegen in Brüche. Die so erhaltene Zahl rundet er wieder auf die nächste ganze Zahl auf, was ihm die Einkaufszahl an Platten ergibt.

Schreiben Sie ein Programm, bei dem Placido die folgenden Eingaben machen kann:

* Breite der Wand
* Breite einer Platte
* Höhe der Wand
* Höhe einer Platte

**Optimierungsmöglichkeit**: Zwischen zwei Platten steht jeweils eine Fuge von 1 bis 3 Millimetern. Placido rechnet diese Breite bei den Platten dazu, um bei großen Wänden jeweils die eine oder andere Spalte (oder Zeile) einsparen zu können. Ihr Programm nimmt zusätzlich die Fugenbreite entgegen und berücksichtigt diese.

**Zusatz**: Stellen Sie die Wand grafisch dar. Jede Platte ist ein #-Zeichen. Die Umrandung wird an den Ecken mit +-Zeichen, horizontale Kanten mit –Zeichen und vertikale Kanten mit |-Zeichen dargestellt.

# Hannas Zahlenschloss

Hannas **fünfstelliges** Zahlenschloss wurde verstellt. Das Zahlenschloss besteht aus fünf Ringen mit je zehn möglichen Einstellungen ('0', '1', '2', ..., '9').

Irgendeine Klassenkameradin hat ihr den Code abgeschaut und nun neu eingestellt.

Die Klassenkameradin hat den neuen Code leider vergessen. Aufgrund von Lieblingszahlen und Mustern haben die beiden zusammen jedoch folgendes rekonstruieren können:

* Der neue Code enthält keine 5.
* Der neue Code enthält mindestens einmal die Ziffer 3.
* Der neue Code enthält mindestens einmal die Ziffer 6.
* Der neue Code startet nicht mit einer geraden Ziffer, auch nicht mit der Null ("0").
* Der neue Code ist **fast** aufsteigend. Das heißt, die nachfolgenden Ziffern sind nicht kleiner als die vorangehenden (z. b. "23368"). Dies jedoch mit **maximal einer Ausnahme**. An einer Stelle darf die Ziffernfolge absteigend sein (z. b. 23326); aber wie erwähnt: maximal einmal!

Lohnt es sich nun, von den theoretisch 100'000 Möglichkeiten die verbleibenden auszuprobieren? Oder soll Hanna doch besser ein neues Zahlenschloss anschaffen.

**Teilaufgabe 1**: Schreiben Sie ein Programm, das alle verbleibenden Möglichkeiten ausgibt.

**Teilaufgabe 2**: Hanna kann eine Kombination in drei (3) Sekunden ausprobieren. Gehen wir davon aus, dass sie die richtige Kombination nach Durchprobieren etwa der Hälfte aller Möglichkeiten gefunden haben wird; wird sie es in weniger als einer Stunde schaffen?

# Es lebe der König!

Ein König nahm bei einem Feldzug 100 feindliche Soldaten gefangen, die er in 100 Einzelzellen sperren ließ. An seinem Geburtstag will er durch Großzügigkeit glänzen und plant eine Amnestie der gefangenen Soldaten.

Allerdings will er nicht alle Soldaten freilassen und fragt deswegen seinen Hofmathematiker, wie er verfahren soll. Dieser antwortet ihm wie folgt: Öffne zunächst alle Türen, schließe sodann jede zweite Türe, öffne nun wieder jede dritte Türe, schließe nun wieder jede vierte Türe und verfahre weiter so bis zum 100. Durchgang.

Der König überlegt eine Weile und sagt dann: Ich habe den Eindruck, bei diesem Verfahren muss ich ziemlich viele Gefangene freilassen. Darauf entgegnet der Hofmathematiker: Nun gut, wenn eure Majestät meinen, dass auf diese Weise zu viele Gefangene freikommen, so habe ich noch ein zweites Verfahren anzubieten:

Öffne zunächst alle Türen, schließe sodann jede zweite Türe, beim dritten Durchgang nehme jede dritte Türe und öffne sie, wenn sie geschlossen ist, bzw. schließe sie, wenn sie geöffnet ist. Im vierten Durchgang wiederum nehme jede vierte Türe und öffne sie, wenn sie geschlossen ist, bzw. umgekehrt. Verfahre nun weiter so bis zum 100. Durchgang. Bei diesem zweiten Verfahren werden weniger Gefangene freikommen.

Bei welchem Verfahren werden wie viel Türen (nach dem 100.Durchgang) offenstehen und welches sind ihre Türnummern. Beachten Sie bitte, dass der König vorm-EDV-Zeitalter lebte. Seine Nummerierung der Türen beginnt natürlich bei 1 und endet bei 100.