Einführung in die Programmierung

Christmas Contest
CC-EidP 2021

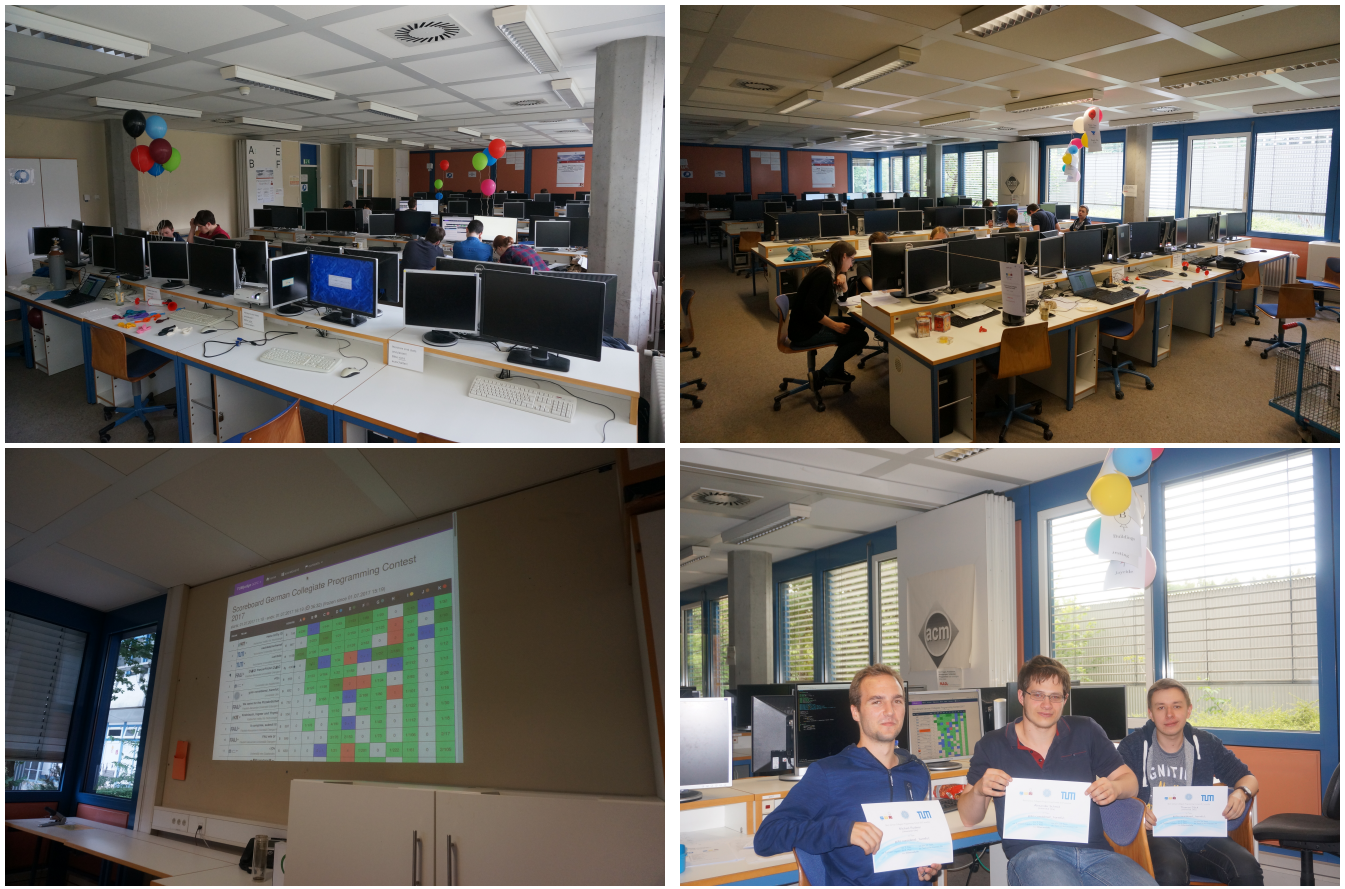
Dr. Gregor Behnke
Wintersemester 2021/2012

The International Collegiate Programming Contest (ICPC)

ICPC

The International Collegiate Programming Contest (ICPC)

- Erste Runde – Deutschland – jedes Jahr im Juni/Juli
  $\rightarrow$ findet an den Unis statt



ICPC

- Zweite Runde – Nord-West-Europa – jedes Jahr im November
  $\rightarrow$ zentral (online), 2–3 Teams pro Uni

ICPC

- Dritte Runde – Welt – jedes Jahr im Sommer
  $\rightarrow$ zentral, 3-4 Teams aus unserer Region

Der „Wintercontest“

- Nullte Runde – Deutschland – jedes Jahr im Januar (29.1.22)
  $\rightarrow$ dezentral (online), mit einfachen Aufgaben!

Weihnachten - Übungsblätter einmal Anders

Statt eines „normalen“ Blatts gibt es über Weihnachten einen Programmierwettbewerb!

Weihnachten - Übungsblätter einmal Anders

Statt eines „normalen“ Blatts gibt es über Weihnachten einen Programmierwettbewerb!

Es gibt 13 nach Schwierigkeit sortierte Aufgaben.

Weihnachten - Übungsblätter einmal Anders

Statt eines „normalen“ Blatts gibt es über Weihnachten einen Programmierwettbewerb!

Es gibt 13 nach Schwierigkeit sortierte Aufgaben.

- 4 Aufgaben á 2.5 Punkte
- 1 Aufgabe á 2 Punkte
- 8 Aufgaben á 1 Punkt

Weihnachten - Übungsblätter einmal Anders

Statt eines „normalen“ Blatts gibt es über Weihnachten einen Programmierwettbewerb!

Es gibt 13 nach Schwierigkeit sortierte Aufgaben.

- 4 Aufgaben á 2.5 Punkte
- 1 Aufgabe á 2 Punkte
- 8 Aufgaben á 1 Punkt

Jeder arbeitet für sich alleine, d.h. Paar- oder Gruppen-Abgaben sind nicht zugelassen.

Bonus-Aufgaben

Alle Aufgaben sind Programmier-Aufgaben.

Bonus-Aufgaben

Alle Aufgaben sind Programmier-Aufgaben.

Sie geben die Aufgaben nicht bei Ihrem Tutor ab, sondern bei einem *Online-Judge* (einem Bewertungssystem).
Der Judge sagt Ihnen direkt, ob Ihre Lösung richtig ist oder nicht.

Bonus-Aufgaben

Alle Aufgaben sind Programmier-Aufgaben.

Sie geben die Aufgaben nicht bei Ihrem Tutor ab, sondern bei einem *Online-Judge* (einem Bewertungssystem).
Der Judge sagt Ihnen direkt, ob Ihre Lösung richtig ist oder nicht.

...

Sie können jede Aufgabe so oft versuchen wie Sie wollen.

Judge

Die Bewertung erfolgt automatisch über unseren Online-Judge, den DOMjudge. Der DOMjudge ist unter der folgen URL erreichbar:

`https://domjudge.informatik.uni-freiburg.de`

Registrieren Sie sich bitte und geben Sie Matrikelnummer an!

Judge

Die Bewertung erfolgt automatisch über unseren Online-Judge, den DOMjudge. Der DOMjudge ist unter der folgen URL erreichbar:

`https://domjudge.informatik.uni-freiburg.de`

Registrieren Sie sich bitte und geben Sie Matrikelnummer an!
Der Wettbewerb wird am

Montag, den 20. Dezember 2021 um 18:00 Uhr

beginnen und am

Montag, den 10. Januar 2022 um 18:00 Uhr

enden.

Judge

Die Bewertung erfolgt automatisch über unseren Online-Judge, den DOMjudge. Der DOMjudge ist unter der folgen URL erreichbar:

`https://domjudge.informatik.uni-freiburg.de`

Registrieren Sie sich bitte und geben Sie Matrikelnummer an!
Der Wettbewerb wird am

Montag, den 20. Dezember 2021 um 18:00 Uhr

beginnen und am

Montag, den 10. Januar 2022 um 18:00 Uhr

enden. Das Scoreboard-Freeze findet (spätestens) statt am

Samstag, den 1. Januar 2022 um 00:00 Uhr

Abgaben sind auch noch nach dem Freeze möglich und Sie erhalten weiterhin Punkte für Ihre Abgaben. Dies sind lediglich den anderen Studierenden nicht mehr gezeigt.

Regeln

- Studierende mit mehr gelösten Aufgaben stehen vor Studierenden mit weniger gelösten Aufgaben.
- Wenn zwei Studierende die selbe Anzahl von korrekten Lösungen haben liegt derjenige mit der kleineren Zeit vorne. Diese Zeit ist die Summe von
  - für jede gelöste Aufgabe: die Anzahl der Minuten, die seit dem Beginn des Wettbewerbs vergangen waren als die erste korrekte Lösung für das Problem eingereicht wurde.
  - für jede falsche Einreichung vor der ersten korrekten Einreichung für eine Aufgabe 1440 Minuten ($\equiv$ 1 Tag).
- Falls auch dies gleich sein sollte, liegt der Studierende vorne, der seine bisher letzte gelöste Aufgabe zuerst gelöst hat.
- Falls auch dies gleich sein sollte, wird rekursiv mit er jeweils zuvor gelösten Aufgabe fortgefahren.

Beispiel-Aufgabe

Gregor ist ein sehr unhöflicher Mensch. Weil ihm das im alltäglichen Leben oft zur Last fällt, versucht er, etwas höflicher zu werden. Als erstes arbeitet er daran, andere Menschen besser zu begrüßen. Bis jetzt sagte er immer, wenn er jemanden traf: „Tach!“.
Eure Aufgabe ist es, Gregor eine höflichere und persönlichere Anrede zu geben.

Eingabe

Die Eingabe beginnt mit der Anzahl T ($1 \leq T \leq 1000$), den Testfällen, also der Anzahl an Personen, die Gregor trifft. Dann folgen T Zeilen, die jeweils einen Namen (ein einziges Wort bestehend aus maximal 20 Buchstaben) enthalten.

Ausgabe

Für jede Person, die Gregor trifft, soll eine Zeile mit dem Wort `Hallo` gefolgt von einem Leerzeichen und dem Namen der Person ausgegeben werden.

Beispieleingabe

```
3
Peter
Simon
Hannes
```

Beispielausgabe

```
Hallo Peter
Hallo Simon
Hallo Hannes
```

Wann ist eine Lösung korrekt

[1] Wir testen `python3 -m py_compile IhrCode.py`

Wann ist eine Lösung korrekt

ACHTUNG!!!

[1] Wir testen `python3 -m py_compile IhrCode.py`

Wann ist eine Lösung korrekt

ACHTUNG!!!

In jeder Aufgabe wird ein bestimmtes Ausgabe-Format beschrieben.
Sie müssen dieses Format exakt einhalten.
Wir haben neben den Beispielen weitere Eingaben, mit denen wir die Korrektheit der Abgaben prüfen.

[1] Wir testen `python3 -m py_compile IhrCode.py`

Wann ist eine Lösung korrekt

ACHTUNG!!!

In jeder Aufgabe wird ein bestimmtes Ausgabe-Format beschrieben.
Sie müssen dieses Format exakt einhalten.
Wir haben neben den Beispielen weitere Eingaben, mit denen wir die Korrektheit der Abgaben prüfen.

Ein Programm erhält **accepted** wenn ihre Ausgabe (bis auf zusätzliche Whitespaces) auf allen Testfällen (insbesondere den geheimen) mit der vorgesehen **exakt** übereinstimmt.

[1] Wir testen `python3 -m py_compile IhrCode.py`

Wann ist eine Lösung korrekt

ACHTUNG!!!

In jeder Aufgabe wird ein bestimmtes Ausgabe-Format beschrieben.
Sie müssen dieses Format exakt einhalten.
Wir haben neben den Beispielen weitere Eingaben, mit denen wir die Korrektheit der Abgaben prüfen.

Ein Programm erhält **accepted** wenn ihre Ausgabe (bis auf zusätzliche Whitespaces) auf allen Testfällen (insbesondere den geheimen) mit der vorgesehen **exakt** übereinstimmt.

Kompiliert Ihr Programm nicht[1], erhalten Sie einen **compiler-error**

[1] Wir testen `python3 -m py_compile IhrCode.py`

Wann ist eine Lösung korrekt

ACHTUNG!!!

In jeder Aufgabe wird ein bestimmtes Ausgabe-Format beschrieben.
Sie müssen dieses Format exakt einhalten.
Wir haben neben den Beispielen weitere Eingaben, mit denen wir die Korrektheit der Abgaben prüfen.

Ein Programm erhält **accepted** wenn ihre Ausgabe (bis auf zusätzliche Whitespaces) auf allen Testfällen (insbesondere den geheimen) mit der vorgesehen **exakt** übereinstimmt.

Kompiliert Ihr Programm nicht[1], erhalten Sie einen **compiler-error**

Stürzt Ihr Programm ab, erhalten Sie einen **run-error**

[1] Wir testen `python3 -m py_compile IhrCode.py`

Wann ist eine Lösung korrekt

ACHTUNG!!!

In jeder Aufgabe wird ein bestimmtes Ausgabe-Format beschrieben.
Sie müssen dieses Format exakt einhalten.
Wir haben neben den Beispielen weitere Eingaben, mit denen wir die Korrektheit der Abgaben prüfen.

Ein Programm erhält **accepted** wenn ihre Ausgabe (bis auf zusätzliche Whitespaces) auf allen Testfällen (insbesondere den geheimen) mit der vorgesehen **exakt** übereinstimmt.

Kompiliert Ihr Programm nicht[1], erhalten Sie einen **compiler-error**

Stürzt Ihr Programm ab, erhalten Sie einen **run-error**

Terminiert Ihr Programm nicht rechtzeitig, erhalten Sie ein **timelimit**

[1] Wir testen `python3 -m py_compile IhrCode.py`

Wann ist eine Lösung korrekt

ACHTUNG!!!

In jeder Aufgabe wird ein bestimmtes Ausgabe-Format beschrieben.
Sie müssen dieses Format exakt einhalten.
Wir haben neben den Beispielen weitere Eingaben, mit denen wir die Korrektheit der Abgaben prüfen.

Ein Programm erhält **accepted** wenn ihre Ausgabe (bis auf zusätzliche Whitespaces) auf allen Testfällen (insbesondere den geheimen) mit der vorgesehen **exakt** übereinstimmt.

Kompiliert Ihr Programm nicht[1], erhalten Sie einen **compiler-error**

Stürzt Ihr Programm ab, erhalten Sie einen **run-error**

Terminiert Ihr Programm nicht rechtzeitig, erhalten Sie ein **timelimit**

Erzeugen Sie falsche Ausgaben, bekommen Sie ein **wrong-answer**

[1] Wir testen `python3 -m py_compile IhrCode.py`

Lösung

```
1   T = int(input())
2
3   for x in range(T):
4       name = input()
5       print("Hallo " + name)
```

Lösung

```
1   T = int(input())
2
3   for x in range(T):
4       name = input()
5       print("Hallo " + name)
```

correct

Lösung

```
1  T = int(input())
2
3  for x in range(T):
4      name = input()
5      print()
6      print(" Hallo   " + name)
7      print()
```

Lösung

```
1   T = int(input())
2
3   for x in range(T):
4       name = input()
5       print()
6       print(" Hallo    " + name)
7       print()
```

correct

Lösung

```
1   T = int(input())
2
3   for x in range(T):
4       print("Gib bitte einen Namen ein!")
5       name = input()
6       print("Hallo " + name)
```

Lösung

```
1   T = int(input())
2
3   for x in range(T):
4       print("Gib bitte einen Namen ein!")
5       name = input()
6       print("Hallo " + name)
```

wrong-answer

Lösung

```
1   T = int(input())
2
3   for x in range(T):
4       name = input()
5       print("Hallo " + name + "!")
```

Lösung

```
1   T = int(input())
2
3   for x in range(T):
4       name = input()
5       print("Hallo " + name + "!")
```

wrong-answer

Lösung

```
1  T = int(input())
2
3  for x in range(T):
4      name = input()
5      print("Hallo " + name)
6
7  while 3 != 4:
8      T+=1
```

Lösung

```
1  T = int(input())
2
3  for x in range(T):
4      name = input()
5      print("Hallo " + name)
6
7  while 3 != 4:
8      T+=1
```

timelimit

Lösung

```
1   T = int(input())
2
3   for x in range(T):
4       name = input()
5       print("Hallo " + name)
6       print(name[-1000])
```

Lösung

```
1  T = int(input())
2
3  for x in range(T):
4      name = input()
5      print("Hallo " + name)
6      print(name[-1000])
```

run-error

Beispiel-Aufgabe

Sabine findet, dass Arrays eine tolle Erfindung sind. Leider weiß sie nicht, wie man zu einem gegebenen Array die Summe aller Elemente des Arrays bestimmt.
Kannst du Sabine dabei helfen?

Eingabe

Die Eingabe beginnt mit der Anzahl T ($1 \leq T \leq 1000$), den Testfällen, auf einer eigenen Zeile. Dann folgen T Testfälle. Jeder Testfall beginnt mit einer Zahl K ($1 \leq K \leq 10000$), die Länge des Arrays. Anschließend folgen K ganze Zahlen Z ($1 \leq Z \leq 100$).

Ausgabe

Für jeden Testfall ist eine einzelne Zeile auszugeben, die die Summe der Werte des Arrays enthält.

Beispieleingabe

```
1
5 1 2 3 4 5
```

Beispielausgabe

```
15
```

Lösung

```
1   T = int(input())
2
3   for x in range(T):
4       zeile = input()
5       elemente = zeile.split()
6
7       summe = 0
8       for i in range(1,len(elemente)):
9           summe += int(elemente[i])
10
11      print(summe)
```

Lösung

```
 1  T = int(input())
 2
 3  for x in range(T):
 4      zeile = input()
 5      elemente = zeile.split()
 6
 7      summe = 0
 8      for i in range(1,len(elemente)):
 9          summe += int(elemente[i])
10
11      print(summe)
```

correct

Technisches

Technisches

- Ihr Code wird mit python3 (Version 3.9.5) ausgeführt. numpy ist installiert und kann verwendet werden.

Technisches

- Ihr Code wird mit python3 (Version 3.9.5) ausgeführt. numpy ist installiert und kann verwendet werden.
- Sie können kein Pattern Matching benutzen (erst ab 3.10 möglich). Das Modul pytest ist nicht installiert.

Technisches

- Ihr Code wird mit python3 (Version 3.9.5) ausgeführt. numpy ist installiert und kann verwendet werden.
- Sie können kein Pattern Matching benutzen (erst ab 3.10 möglich). Das Modul pytest ist nicht installiert.
- Es gibt eine *angemessene* Speicherbeschränkung

Technisches

- Ihr Code wird mit python3 (Version 3.9.5) ausgeführt. numpy ist installiert und kann verwendet werden.
- Sie können kein Pattern Matching benutzen (erst ab 3.10 möglich). Das Modul pytest ist nicht installiert.
- Es gibt eine *angemessene* Speicherbeschränkung
- Ihr Programm liest alle Eingaben von der Standard-Eingabe (`input`)

Technisches

- Ihr Code wird mit python3 (Version 3.9.5) ausgeführt. numpy ist installiert und kann verwendet werden.
- Sie können kein Pattern Matching benutzen (erst ab 3.10 möglich). Das Modul pytest ist nicht installiert.
- Es gibt eine *angemessene* Speicherbeschränkung
- Ihr Programm liest alle Eingaben von der Standard-Eingabe (`input`)
- Ihr Programm schreibt alle Ausgaben in die Standard Ausgabe (`print`)

Technisches

- Ihr Code wird mit python3 (Version 3.9.5) ausgeführt. numpy ist installiert und kann verwendet werden.
- Sie können kein Pattern Matching benutzen (erst ab 3.10 möglich). Das Modul pytest ist nicht installiert.
- Es gibt eine *angemessene* Speicherbeschränkung
- Ihr Programm liest alle Eingaben von der Standard-Eingabe (`input`)
- Ihr Programm schreibt alle Ausgaben in die Standard Ausgabe (`print`)
- Beendet sich ihr Programm nicht mit dem Exit-Code 0, wird dies als **run-error** gewertet.

Technisches

- Ihr Code wird mit python3 (Version 3.9.5) ausgeführt. numpy ist installiert und kann verwendet werden.
- Sie können kein Pattern Matching benutzen (erst ab 3.10 möglich). Das Modul pytest ist nicht installiert.
- Es gibt eine *angemessene* Speicherbeschränkung
- Ihr Programm liest alle Eingaben von der Standard-Eingabe (`input`)
- Ihr Programm schreibt alle Ausgaben in die Standard Ausgabe (`print`)
- Beendet sich ihr Programm nicht mit dem Exit-Code 0, wird dies als **run-error** gewertet.
- Neben Python erlauben wir auch Abgaben in: C, C++, Java, C#, Haskell, Go. Weitere Programmiersprachen können bei vertretbarem Aufwand hinzugefügt werden.

Technisches

- Ihr Code wird mit python3 (Version 3.9.5) ausgeführt. numpy ist installiert und kann verwendet werden.
- Sie können kein Pattern Matching benutzen (erst ab 3.10 möglich). Das Modul pytest ist nicht installiert.
- Es gibt eine *angemessene* Speicherbeschränkung
- Ihr Programm liest alle Eingaben von der Standard-Eingabe (`input`)
- Ihr Programm schreibt alle Ausgaben in die Standard Ausgabe (`print`)
- Beendet sich ihr Programm nicht mit dem Exit-Code 0, wird dies als **run-error** gewertet.
- Neben Python erlauben wir auch Abgaben in: C, C++, Java, C#, Haskell, Go. Weitere Programmiersprachen können bei vertretbarem Aufwand hinzugefügt werden.
- Versuche, Dateien einzulesen oder auszugeben, neue Threads zu starten oder Netzwerkverbindungen zu öffnen, etc. werden als Betrugsversuch gewertet.

Technisches

- Stellen Sie Fragen immer über das Clarification-System des DOMjudge.

Technisches

- Stellen Sie Fragen immer über das Clarification-System des DOMjudge.
- Bei einigen Problemen kann es einige Zeit (mehrere Minuten) dauern, bis der DOMjudge auf ihre Abgabe antwortet.

Technisches

- Stellen Sie Fragen immer über das Clarification-System des DOMjudge.
- Bei einigen Problemen kann es einige Zeit (mehrere Minuten) dauern, bis der DOMjudge auf ihre Abgabe antwortet.
- In vergangenen Versionen des DOMjudge haben Umlaute o.ä. (auch in Kommentaren) zu (Compile-)Fehlern geführt. Dieses Problem wurde in der aktuellen Version teilweise behoben, solange sie UTF-8 Umlaute benutzten. In Java könnten Klassennamen die Umlaute enthalten in einigen Fällen immer noch zu Problemen führen. Versuchten Sie bitte dies zu vermeiden.

Tipps

Tipps

- Lesen Sie die Aufgaben gründlich. Manchmal sind Details von großer Wichtigkeit!

Tipps

- Lesen Sie die Aufgaben gründlich. Manchmal sind Details von großer Wichtigkeit!
- Sie können sich darauf verlassen, dass alle Eingaben so sind, wie wir es in der Aufgabe spezifiziert haben. Sie brauchen dies nicht in ihrem Programm zu testen und Sie brauchen keine Fehlerbehandlung zu schreiben.

Tipps

- Lesen Sie die Aufgaben gründlich. Manchmal sind Details von großer Wichtigkeit!
- Sie können sich darauf verlassen, dass alle Eingaben so sind, wie wir es in der Aufgabe spezifiziert haben. Sie brauchen dies nicht in ihrem Programm zu testen und Sie brauchen keine Fehlerbehandlung zu schreiben.
- Es gibt in der Regel Testfälle mit den in der Aufgabe angegebenen Maximalgrößen. Stellen Sie sich darauf ein, dass Sie diese lösen müssen.

Tipps

- Lesen Sie die Aufgaben gründlich. Manchmal sind Details von großer Wichtigkeit!
- Sie können sich darauf verlassen, dass alle Eingaben so sind, wie wir es in der Aufgabe spezifiziert haben. Sie brauchen dies nicht in ihrem Programm zu testen und Sie brauchen keine Fehlerbehandlung zu schreiben.
- Es gibt in der Regel Testfälle mit den in der Aufgabe angegebenen Maximalgrößen. Stellen Sie sich darauf ein, dass Sie diese lösen müssen.
- Erzeugen Sie exakt die beschriebenen Ausgaben.

Tipps

- Lesen Sie die Aufgaben gründlich. Manchmal sind Details von großer Wichtigkeit!
- Sie können sich darauf verlassen, dass alle Eingaben so sind, wie wir es in der Aufgabe spezifiziert haben. Sie brauchen dies nicht in ihrem Programm zu testen und Sie brauchen keine Fehlerbehandlung zu schreiben.
- Es gibt in der Regel Testfälle mit den in der Aufgabe angegebenen Maximalgrößen. Stellen Sie sich darauf ein, dass Sie diese lösen müssen.
- Erzeugen Sie exakt die beschriebenen Ausgaben.
- Erst überlegen – dann programmieren. Manchmal ist die Lösung deutlich einfacher als man denkt.

Tipps

- Lesen Sie die Aufgaben gründlich. Manchmal sind Details von großer Wichtigkeit!
- Sie können sich darauf verlassen, dass alle Eingaben so sind, wie wir es in der Aufgabe spezifiziert haben. Sie brauchen dies nicht in ihrem Programm zu testen und Sie brauchen keine Fehlerbehandlung zu schreiben.
- Es gibt in der Regel Testfälle mit den in der Aufgabe angegebenen Maximalgrößen. Stellen Sie sich darauf ein, dass Sie diese lösen müssen.
- Erzeugen Sie exakt die beschriebenen Ausgaben.
- Erst überlegen – dann programmieren. Manchmal ist die Lösung deutlich einfacher als man denkt.
- Testen Sie Ihr Programm immer auf den Beispiel-Eingaben.

Tipps

Tipps

- Alle Aufgaben haben kurze Lösungen. Jede Aufgabe ist mit weniger als 100 Zeilen lösbar. Die einfachen sogar mit viel weniger. Die kürzeste Lössung ist 2, die Längste ist nur 57 Zeilen lang. Sie können alle Aufgaben mit insgesamt 334 Zeilen Code lösen.

Tipps

- Alle Aufgaben haben kurze Lösungen. Jede Aufgabe ist mit weniger als 100 Zeilen lösbar. Die einfachen sogar mit viel weniger. Die kürzeste Lössung ist 2, die Längste ist nur 57 Zeilen lang. Sie können alle Aufgaben mit insgesamt 334 Zeilen Code lösen.
- Die Aufgaben sind in aufsteigender Schwierigkeit sortiert. Wenn Sie eine Aufgabe nicht lösen können kann es aber gut sein, dass Sie die nächste schaffen.

Tipps

- Alle Aufgaben haben kurze Lösungen. Jede Aufgabe ist mit weniger als 100 Zeilen lösbar. Die einfachen sogar mit viel weniger. Die kürzeste Lössung ist 2, die Längste ist nur 57 Zeilen lang. Sie können alle Aufgaben mit insgesamt 334 Zeilen Code lösen.
- Die Aufgaben sind in aufsteigender Schwierigkeit sortiert. Wenn Sie eine Aufgabe nicht lösen können kann es aber gut sein, dass Sie die nächste schaffen.
- Bei den schwereren Aufgaben ist die Komplexität Ihres Algorithmus relevant. Sie können dabei folgende Daumenregel verwenden:

  Setzen Sie die maximalen Größen aller Eingaben in die $\mathcal{O}$-Formel ein. Wenn deutlich mehr als 1.000.000 herauskommt wird ihr Programm zu langsam sein.

Eine Warnung

Eine Warnung

Nach dem Contest werden alle Abgaben auf Plagiate überprüft. Sollten wir eines finden, verfallen für alle Beteiligten (Abschreiber und Abschreibenlasser) **alle** Punkte.

Eine Warnung

Nach dem Contest werden alle Abgaben auf Plagiate überprüft.
Sollten wir eines finden, verfallen für alle Beteiligten (Abschreiber und Abschreibenlasser) **alle** Punkte.
Das selbe gilt für jegliche Art von Betrugsversuchen.

Ich wünschen Ihnen frohe Weihnachten!

Ich wünschen Ihnen frohe Weihnachten!

... und viel Spaß beim programmieren!