40. Bundeswettbewerb Informatik 2021/2022

Die Aufgaben der 2. Runde

Bundeswettbewerb Informatik

Allgemeine Hinweise

Herzlichen Glückwunsch zum Erreichen der 2. Runde! Hier sind die Aufgaben. Sie sind anspruchsvoll, und ihre Bearbeitung ist aufwendig. Aber die Mühe lohnt sich, denn durch Teilnahme an der 2. Runde

- wirst du sicher sehr viel lernen;
- kannst du dich für die Endrunde qualifizieren;
- kannst du einen Buchpreis der Verlage O'Reilly oder dpunkt.verlag gewinnen;
- hast du am Ende eine Arbeit fertig gestellt, die du als Besondere Lernleistung in die Abiturwertung einbringen kannst;
- kannst du dich (als jüngerer Teilnehmer) um die Teilnahme an einer Deutschen Schülerakademie bewerben;
- hast du die Chance auf eine Einladung zu den „Forschungstagen Informatik 2022" des Max-Planck-Instituts für Informatik in Saarbrücken.

Wir wünschen also viel Spaß und viel Erfolg bei der Bearbeitung!

Es gibt drei Aufgaben – und aus Anlass des Jubiläums (40. Wettbewerb) zusätzlich eine Bonusaufgabe, als besondere Herausforderung. **Eine Einsendung darf Bearbeitungen zu höchstens zwei dieser insgesamt vier Aufgaben enthalten**, deren Bewertung dann das Gesamtergebnis ausmacht. Sollte eine Einsendung Bearbeitungen zu mehr als zwei Aufgaben enthalten, werden wir zwei davon zufällig auswählen und nur diese bewerten.

An dieser Runde dürfen nur Einzelpersonen teilnehmen, die in der 1. Runde in drei Aufgaben insgesamt mindestens 12 Punkte erreicht oder einem Team angehört haben, dem dieses gelungen ist. Gruppenarbeit ist in der 2. Runde nicht zulässig.

Einsendeschluss ist Montag, der 25. April 2022.

Bearbeitung

Die Bearbeitung einer Aufgabe sollte zunächst eine nachvollziehbare und vollständige Lösung aller Teilaufgaben enthalten. **Zusatzpunkte** für eine höhere Bewertung kannst du erreichen, wenn du die Aufgabe dort, wo es möglich und sinnvoll ist, eigenständig weiterentwickelst. Sinnvoll sind inhaltliche Erweiterungen und Verbesserungen, etwa von Datenstrukturen und Algorithmen, die praktisch realisiert werden; uninteressant sind aufwendige Tricks, z. B. zur reinen Verschönerung der Benutzungsoberfläche. Begründe für jede Erweiterung, weshalb sie sinnvoll ist und ihre Realisierung eine eigene Schwierigkeit darstellt.

Grundsätzlich gelten die Vorgaben der 1. Runde weiter. Wesentliches Ergebnis der Aufgabenbearbeitung ist also eine **Dokumentation**, in der du den *Lösungsweg* sowie die *Umsetzung* des Lösungswegs in das dazugehörige Programm beschreibst. Die Beschreibung des Lösungswegs

kann mit Hilfe (halb-)formaler Notationen präzisiert werden, die Beschreibung der Umsetzung mit Verweisen auf die entsprechenden Quellcode-Elemente.

In die Dokumentation gehören auch aussagekräftige *Beispiele* (Programmeingaben/-ausgaben, ggf. inklusive Zwischenschritte/-ergebnisse), die zeigen, wie das Programm sich in unterschiedlichen Situationen verhält. Komplettiert wird die Dokumentation durch *Auszüge aus dem Quelltext*, die alle wichtigen Module, Methoden, Funktionen usw. enthalten. Die Beschreibung des Lösungswegs und der Umsetzung sollte jedoch keinen oder nur wenig Quellcode enthalten.

Weiteres Ergebnis der Aufgabenbearbeitung ist die **Implementierung**. Sie besteht aus dem zur Lösung der Aufgabe geschriebenen lauffähigen *Programm* und dem vollständigen *Quelltext*. Außerdem können Beispieleingabe/-ausgaben oder weiteres hilfreiches Material der Implementierung beigefügt werden.

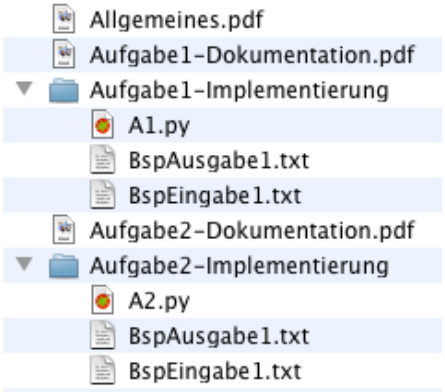
Die Dokumentation zu einer Aufgabe mit allen oben genannten Bestandteilen muss als PDF-Dokument eingereicht werden. **Es kann sein, dass für die Bewertung deiner Einsendung nur die Dokumentation herangezogen wird.** Sie sollte also einen lückenlosen und verständlichen Nachweis des Leistungsumfangs und der Funktionstüchtigkeit der Programme geben – und unbedingt die vorgegebenen Beispiele neben eigenen enthalten!

Der Umfang der Dokumentation soll sich in Grenzen halten; eine gute Dokumentation vermittelt kurz und präzise alles Nötige, insbesondere die wesentlichen Ideen beim Lösungsweg. Nötig ist alles, was Interessierte mit guten Informatikkenntnissen, die die Aufgabenstellung kennen, wissen müssen, um den Lösungsweg zu verstehen und seine Umsetzung nachzuvollziehen.

Entscheidend für eine gute Bewertung sind zwar richtige (und sauber umgesetzte) Lösungswege, aber die Dokumentation hat schon oft den Ausschlag für oder gegen das Weiterkommen gegeben. Das Erstellen der Dokumentation sollte die Arbeit an Lösungsideen und Umsetzung eng begleiten. Wer zunächst die Lösungsidee verständlich formuliert, dem fällt anschließend eine fehlerlose Implementierung leichter. Abbildungen tragen in der Regel zur Verständlichkeit bei, und es schadet nicht, die Dokumentation von Dritten prüfen zu lassen, selbst wenn diese fachfremd sind.

Einsendung

Die Einsendung erfolgt wieder über das BWINF AMS (login.bwinf.de). Hochladen kannst du ein max. 40 MB großes ZIP-Archiv (z. B. `VornameNachname.zip`). Sein Inhalt sollte so strukturiert sein wie rechts abgebildet. Die Dokumentationen der bearbeiteten Aufgaben müssen als PDF-Dokumente enthalten sein; Dateien in anderen Formaten werden möglicherweise ignoriert. Ein Dokument `Allgemeines.pdf` ist nur dann nötig, wenn du allgemeine, von den Aufgabenbearbeitungen unabhängige Bemerkungen zu deiner Einsendung machen willst. Die Schriftgröße einer Dokumentation muss mindestens 10 Punkt sein, bei Quelltext mindestens 8 Punkt. Auf jeder Seite einer Dokumentation sollen in der Kopfzeile die Teilnahme-ID, Vorname, Name und Seitennummer stehen; hierfür sind auf den BWINF-Webseiten Vorlagen zu finden. Die Teilnahme-



ID steht auf der Teilnahmebescheinigung der 1. Runde, und du findest sie auch im AMS; es handelt sich um eine Zahl zwischen 59.000 und 65.000.

Weitere Hinweise

Bei der Bewertung können Programme unter Windows (10), Linux, Mac OS X (12.1) und Android ausgeführt werden.

Fragen zu den Aufgaben können per Mail an `bundeswettbewerb@bwinf.de` oder, zu üblichen Arbeitszeiten, telefonisch unter 0228 378646 gestellt werden. Die Antwort auf E-Mail-Anfragen kann sich leicht verzögern. Informationen zur 2. Runde finden sich auf unseren Webseiten: `bwinf.de/bundeswettbewerb`. Unter `einstieg-informatik.de` findest du unsere Community; dort werden im Forum sicher wieder viele Teilnehmerinnen und Teilnehmer über die Aufgaben diskutieren – ohne Lösungsideen auszutauschen.

Allen Teilnehmern der 2. Runde wird bis Mitte Juni 2022 die Bewertung mitgeteilt. Die Besten werden zur Endrunde eingeladen, die 13.-16. September 2022 von den Trägern der Bundesweiten Informatikwettbewerbe ausgerichtet wird: Gesellschaft für Informatik e.V., Fraunhofer-Verbund IUK-Technologie und Max-Planck-Institut für Informatik. Dort wird entschieden, wer mit einem Bundessieg oder mit einem zweiten Preis ausgezeichnet wird. Bundessiegerinnen und Bundessieger werden in der Regel ohne weiteres Auswahlverfahren in die Förderung der Studienstiftung des deutschen Volkes aufgenommen. Außerdem werden Geldpreise vergeben. Der Rechtsweg ist ausgeschlossen.

Viel Spaß und viel Erfolg!

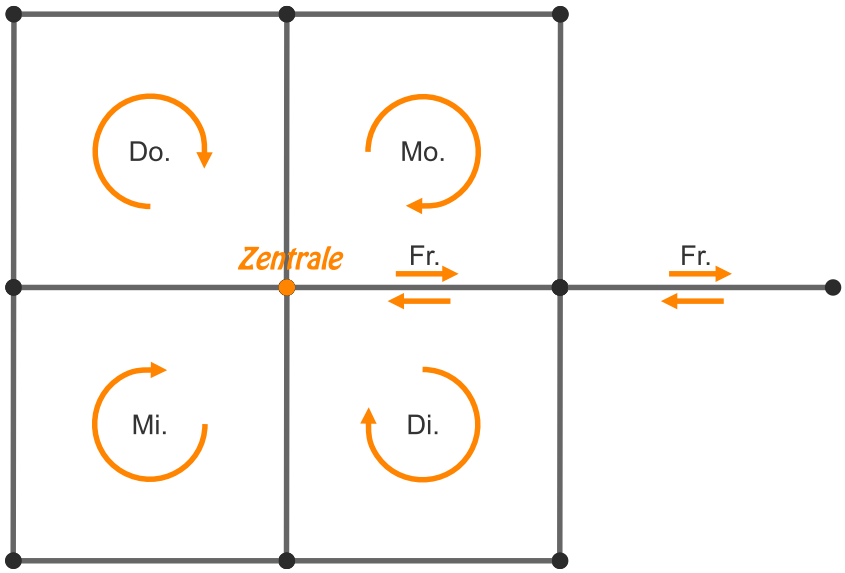
Aufgabe 1: Müllabfuhr

Aufstand in Müllhausen! Die städtische Müllabfuhr CityTrash, die leider nur über genau ein Müllauto verfügt, erhält immer mehr Beschwerden von Anwohnern, dass an deren Straßen die Mülleimer nicht oft genug geleert werden. Um Abhilfe zu schaffen, soll jetzt jede Straße mindestens einmal pro Woche abgefahren werden.

Hierzu soll ein Wochenfahrplan erstellt werden, der für jeden Wochentag zwischen Montag und Freitag festlegt, welche Straßen das Müllauto abfahren soll. Dabei kann eine Straße an einem Tag auch mehrmals befahren werden. Da die Straßen von Müllhausen sehr eng sind, kann der Fahrer nur an Kreuzungen oder am Ende von Sackgassen wenden; selbst Rückwärtsfahren ist zu gefährlich. Jede Tagesroute beginnt und endet in der Zentrale von CityTrash.

Um Überstunden zu vermeiden, soll die maximale Tagesstrecke möglichst klein gehalten werden.

Ein Beispiel: Wenn Müllhausen so einen einfachen Straßenplan hätte wie unten im Bild und alle Straßen 1 km lang wären, dann kann das Müllauto von Montag bis Donnerstag jeweils einen der „Straßenquadranten“ und am Freitag die beiden Straßen nach Osten und zurück abfahren. Alle Tagesstrecken und somit auch die maximale Tagesstrecke sind 4 km lang. Dieser Wochenfahrplan ist optimal: Wenn das Müllauto an einem Tag eine kürzere Strecke fahren würde, müsste es an mindestens einem anderen Tag eine längere Strecke abfahren; die maximale Tagesstrecke wäre dann mehr als 4 km lang.



Aufgabe

Hilf CityTrash, indem du ein Programm schreibst, das eine Karte einliest und einen passenden Wochenfahrplan erzeugt. Auf der Karte sind die Straßen von Müllhausen und die Zentrale von CityTrash eingezeichnet.

Wende dein Programm mindestens auf alle Beispiele an, die du auf den BWINF-Webseiten findest, und dokumentiere die Ergebnisse.

Aufgabe 2: Rechenrätsel

Gwendoline stellt ihren Mitschülern gerne Rätsel der Form

$$4 \circ 4 \circ 3 = 13.$$

Für „$\circ$“ sind die Operatoren $+, -, *, :$ (Addition, Subtraktion, Multiplikation, Division) einzusetzen, so dass die Rechnung stimmt. Im obigen Beispiel wäre $4 * 4 - 3 = 13$ die richtige Lösung.

Du möchtest dich revanchieren und Gwendoline richtig schwere Rätsel dieser Art stellen.

Aufgabe

Schreibe ein Programm, das Rätsel nach Gwendolines Schema erstellt. Die Rätsel sollten interessant und unterschiedlich sein (also z.B. nicht $2 \circ 2 \circ 2 \circ 2 = 16$). Das Programm sollte für eine gegebene Anzahl von Operatoren ein Rätsel so erzeugen, dass

a) das Rätsel eindeutig lösbar ist (also nicht $3 \circ 4 \circ 3 = 15$, welches zwei Lösungen hat),

b) die Operanden einzelne Ziffern sind,

c) eine positive ganze Zahl als Ergebnis herauskommt,

d) Punkt- vor Strich-Rechnung angewandt wird,

e) gleichrangige Operatoren linksassoziativ angewandt werden (also z.B. $6 : 3 * 2 = 4$ oder $5 - 2 + 1 = 4$) und

f) alle Zwischenergebnisse ganze Zahlen sind (also nicht $3 : 2 * 4 = 6$).

Lasse dein Programm für verschiedene Operatorenanzahlen laufen und zeige uns mehrere deiner Ergebnisse.

Schaffst du es, Rätsel mit 10, 15 oder sogar noch mehr Operatoren zu generieren? Hier ist so ein Rätsel:

$$4 \circ 3 \circ 2 \circ 6 \circ 3 \circ 9 \circ 7 \circ 8 \circ 2 \circ 9 \circ 4 \circ 4 \circ 6 \circ 4 \circ 4 \circ 5 = 4792$$

Aufgabe 3: Hex-Max

Severin hat heute in der Schule das Hexadezimalsystem kennengelernt. Er verwendet nun kurze Stäbchen, um die sechzehn Ziffern darzustellen, wie in einer Siebensegmentanzeige. Für die Darstellung einer Ziffer stehen also sieben Positionen zur Verfügung; jede Position ist entweder durch ein Stäbchen belegt oder sie ist frei.

Severin will nun das folgende, von der Lehrerin gestellte Rätsel lösen: Gegeben ist eine Hexadezimalzahl (kurz: Hex-Zahl) mit n Ziffern. Gesucht ist die größte Hex-Zahl mit der gleichen Anzahl an Ziffern, die aus der gegebenen Zahl durch eine zusätzlich gegebene Maximalzahl an Umlegungen erzeugt werden kann. „Umlegung“ bedeutet, ein Stäbchen von seiner bisherigen Position an eine andere, bislang freie Position zu bewegen.

Eine einzelne Umlegung muss noch keine gültige Hex-Zahl ergeben, aber das Ergebnis nach allen Umlegungen muss eine gültige Hex-Zahl in der Siebensegmentdarstellung sein. Zu keiner Zeit darf die Darstellung einer Ziffer komplett „geleert“ werden. Die gegebene Maximalzahl an Umlegungen muss nicht ausgeschöpft werden.

Hier ein Beispiel: Gegeben sind die Zahl D24 und die Maximalzahl an Umlegungen 3. Die gesuchte Hex-Zahl ist EE4.

Aufgabe

Schreibe ein Programm, das eine Hex-Zahl sowie die Maximalzahl m an Umlegungen einliest und die größte Hexadezimalzahl ermittelt, die mit höchstens m Umlegungen erzeugt werden kann. Das Programm soll nach jeder Umlegung den Zwischenstand, also die aktuelle Belegung der Positionen ausgeben.

Wende dein Programm mindestens auf alle Beispiele an, die du auf den BWINF-Webseiten findest, und dokumentiere die Ergebnisse.

Bonusaufgabe: Zara Zackigs Zurückkehr

Zara Zackig hat inzwischen große Karriere gemacht und ihre eigene Informatikfirma aufgebaut, wobei sie ihre Leidenschaft zur Kryptographie erfolgreich einsetzen konnte. Diese beschäftigt sie auch privat, was ihr folgendes Problem bereitete:

Zara besitzt jetzt zehn Ferienhäuser die sie zyklisch jedes Wochenende aufsucht. Zur Zugangskontrolle verwendet sie ein selbstentworfenes System, das Lochkarten mit 128 Positionen verwendet. Jede Lochkarte kann also ein beliebiges 128 Bits umfassendes Codeword darstellen.

Für ihre zehn Häuser stellte sie zehn Karten mit zufällig generierten Codewörten her, sortierte sie anschließend aufsteigend als $w_1, \ldots, w_{10}$ und codierte die Schlösser der zehn Häuser, wobei Haus k das Codewort w_k erhielt.

Nach einiger Zeit bekam Zara aber Angst, dass eine Karte auf dem Weg verloren gehen könnte. Die meisten Menschen hätten jetzt wahrscheinlich einfach Kopien der Karten an einem sicheren Platz hinterlegt. Nicht so Zara! Sie stellte nur *eine* zusätzliche Karte her, deren Bitmuster das exklusive Oder aller zehn Karten enthält. Mithilfe dieser Sicherungskarte kann sie jetzt im Notfall eine verlorene Karte rekonstruieren.

So weit so gut. Leider sind Zaras Freunde ebenso verrückt wie sie selbst und spielten ihr einen Streich: Sie erstellten weitere 100 Karten mit zufälligen Codewörtern und mischten sie mit Zaras elf Karten in einem großen Stapel, den dann Zara so vorfand. Dummerweise sind die elektronischen Schlösser in Zaras Häusern so konstruiert, dass sie nach drei fehlerhaften Zugangsversuchen das Haus für 24 Stunden verriegeln. Zara konnte das Dilemma allerdings lösen indem sie ein Programm schrieb, das die 111 Karten einlas und dann herausfand, welche 11 Karten die echten waren.

Aufgabe

a) Tu es Zara gleich und schreibe ein Programm, das diese Aufgabenstellung bewältigt. Wende es auf die 111 Karten an, die du auf den BWINF-Webseiten findest. Welche Karten sind die richtigen und welche wurden durch die „Freunde“ hinzugefügt?

b) Wie kann nun Zara mithilfe der 11 gefundenen Karten am nächsten Wochenende das nächste Haus aufsperren, ohne dafür mehr als zwei Fehlversuche zu benötigen?

c) Auf den BWINF-Webseiten findest du weitere Beispielaufgaben, an denen du dein Können demonstrieren kannst. Einige davon sind sehr schwer!