

22. Bundeswettbewerb Informatik

Einsendeschluss:
17. November 2003

Das Aufgabenblatt

www.bwinf.de

Gesellschaft für Informatik e.V.

Die Gesellschaft für Informatik e.V. (GI) ist mit rund 22.500 Mitgliedern die größte Fachgesellschaft im Bereich Informatik. Ihre Mitglieder kommen aus allen Sparten der Wissenschaft, der Informatikindustrie, aus dem Kreis der Anwender sowie aus Lehre, Studium und Ausbildung. In der GI wirken Männer und Frauen am Fortschritt der Informatik mit, vor allem im wissenschaftlich-fachlichen Austausch in etwa 120 verschiedenen Fachgruppen und 35 Regionalgruppen. Ihr gemeinsames Ziel ist die Förderung der Informatik in Forschung, Lehre und Anwendung, die gegenseitige Unterstützung bei der Arbeit sowie die Weiterbildung. Die GI vertritt hierbei die Interessen der Informatik in Politik und Wirtschaft. Im Web: <http://www.gi-ev.de>

Fraunhofer-Gesellschaft, IuK-Gruppe

Die Fraunhofer-Gesellschaft zur Förderung der Angewandten Wissenschaften e.V. führt mit breitem thematischen Spektrum anwendungsorientierte Forschung und Entwicklung durch. Sie kooperiert dabei mit nationalen und internationalen Partnern aus Wirtschaft und Wissenschaft. Mit 56 Instituten und mehr als 11.000 Mitarbeitern ist die Fraunhofer-Gesellschaft eine der größten Forschungsinstitutionen in Europa.

Die informations- und kommunikationstechnischen Institute der Fraunhofer-Gesellschaft sind zur Fraunhofer IuK-Gruppe zusammengeschlossen. Die Fraunhofer IuK-Gruppe koordiniert die anwendungsnahe Forschung und Vorlaufforschung der 15 Mitgliedsinstitute mit zusammen mehr als 2.400 Mitarbeitern und einem Gesamtbudget von über 200 Millionen Euro. Die IuK-Gruppe ist nach Volumen und thematischer Breite der größte Forschungsverbund für Informations- und Kommunikationstechnik in Europa und einer der größten der Welt. Im Web: <http://www.iuk.fhg.de>



Unter der Schirmherrschaft des Bundespräsidenten



Bundesministerium
für Bildung
und Forschung



Gesellschaft für Informatik e.V.



Fraunhofer
Gruppe
Informations- und
Kommunikationstechnik

Träger des Wettbewerbs:
Bundesministerium für Bildung und Forschung,
Gesellschaft für Informatik e.V. (GI),
Fraunhofer-Gesellschaft

Bundeswettbewerb Informatik

Der Bundeswettbewerb Informatik wurde 1980 von der Gesellschaft für Informatik e.V. (GI) auf Initiative von Prof. Dr. Volker Claus ins Leben gerufen. Ziel des Wettbewerbs ist es, Interesse an der Informatik zu wecken und zu intensiver Beschäftigung mit ihren Inhalten und Methoden sowie den Perspektiven ihrer Anwendung anzuregen. Er gehört zu den bundesweiten Schülerwettbewerben, die von den Kultusministern der Länder unterstützt werden. Gefördert wird er vom Bundesministerium für Bildung und Forschung und steht unter der Schirmherrschaft des Bundespräsidenten. Die Träger des Wettbewerbs sind die GI und die Fraunhofer-Gesellschaft. Die Gestaltung des Wettbewerbs und die Auswahl der Sieger obliegen dem Beirat; Vorsitzender: Prof. Dr. Uwe Schöning, Universität Ulm. Die Auswahl und Entwicklung von Aufgaben und die Festlegung von Bewertungsverfahren übernimmt der Aufgabenausschuss. Die Geschäftsstelle des Wettbewerbs ist in Bonn und ist für die fachliche und organisatorische Durchführung zuständig; Geschäftsführer: Dr. Wolfgang Pohl.

→ Start und Ziel im September

Der Wettbewerb beginnt und endet im September, dauert etwa ein Jahr und besteht aus drei Runden. In der ersten und zweiten Runde sind fünf bzw. drei Aufgaben zu Hause selbstständig zu bearbeiten. Dabei können die Aufgaben der ersten Runde mit grundlegenden Informatikkenntnissen gelöst werden; die Aufgaben der zweiten Runde sind deutlich schwieriger. In der ersten Runde ist Gruppenarbeit zugelassen und erwünscht. An der zweiten Runde dürfen jene teilnehmen, die allein oder zusammen mit anderen wenigstens drei Aufgaben weitgehend richtig gelöst haben. In der zweiten Runde ist dann eigenständige Einzelarbeit gefordert; die Bewertung erfolgt durch eine relative Platzierung der Arbeiten. Die ca. dreißig bundesweit Besten werden zur dritten Runde, einem Kolloquium, eingeladen. Darin führt jeder ein Gespräch mit einem Informatiker aus Schule und Hochschule und analysiert und bearbeitet im Team zwei Informatik-Probleme.

→ Wer ist teilnahmeberechtigt?

Teilnehmen können Jugendliche, die nach dem 17.11.1981 geboren wurden. Sie dürfen jedoch zum 1.9.2003 noch nicht ihre (informatikbezogene) Ausbildung abgeschlossen oder eine Berufstätigkeit aufgenommen haben. Ebenfalls ausgeschlossen sind Personen, die zum Wintersemester 2003/2004 oder früher ihr Studium an einer Hochschule/Fachhochschule aufnehmen bzw. aufgenommen haben. Jugendliche, die nicht deutsche Staatsangehörige sind, müssen wenigstens vom 1.9. bis 17.11.2003 ihren Wohnsitz in Deutschland haben oder eine staatlich anerkannte deutsche Schule im Ausland besuchen.

→ Als Anerkennung ...

In allen Runden des Wettbewerbs wird die Teilnahme durch eine Urkunde bestätigt. In der ersten Runde werden darüber hinaus erste und zweite Preise sowie Anerkennungen vergeben; mit einem Preis ist die Qualifikation für die zweite Runde verbunden. Auch in der zweiten Runde gibt es erste und zweite Preise; mit einem zweiten Preis ist ein Sachpreis verbunden. Die Gewinner eines ersten Preises in der zweiten Runde werden zur dritten Runde eingeladen, die im Herbst 2004 stattfinden wird. Die dort ermittelten Bundessieger werden in der Regel ohne weiteres Aufnahmeverfahren in die Studienstiftung des deutschen Volkes aufgenommen. Zusätzlich sind für den Bundessieger, aber auch für andere besondere Leistungen Geld- und Sachpreise vorgesehen, unter anderem ein Aufenthalt an einer Sommerschule im Ausland sowie Einladungen zu Schülerakademien in Deutschland.

→ ... Teilnahme an der Informatik-Olympiade

Ausgewählte Endrundenteilnehmer können sich in mehreren Trainingsrunden für das vierköpfige deutsche Team qualifizieren, das an der Internationalen Olympiade in Informatik 2005 in Polen teilnimmt.

→ ... Informatik-Seminare

Für erfolgreiche BWINF-Teilnehmer aus Baden-Württemberg wird Anfang 2004 erneut das "Jugendforum Informatik" auf der Burg Liebenzell vom Kultusministerium des Landes durchgeführt. Es gibt viel zu lernen, und als Highlight des einwöchigen Informatikworkshops steht ein Besuch bei IBM auf dem Programm. Bei der eintägigen Veranstaltung im IBM-Entwicklungslabor in Böblingen geht es um technologische Zukunftsthemen. Dazu kommen Einblicke in die IT-Trends von morgen und viele nützliche Informationen über das Engagement von IBM in Sachen Ausbildung. Ein weiteres Seminar, das mit Unterstützung der Klaus-Tschira-Stiftung stattfinden würde, ist noch in Planung.



**IBM und SuSE wünschen allen
Teilnehmern des 22. Bundeswettbewerbs
Informatik viel Erfolg!**



Preise beim 22. BWINF – Teilnehmer werben Teilnehmer

Auch beim 22. BWINF gibt es wieder etwas zu gewinnen. 10 aktuelle Versionen von SuSE Linux werden vergeben, und zwar u.a. an die besten Erstrandteilnehmer sowie an Lehrkräfte und Schulen mit besonders vielen Teilnehmern. Der Firma SuSE sei herzlich gedankt.

Unter dem Motto Teilnehmer werben Teilnehmer wollen wir wieder versuchen, den Austausch zwischen erfahrenen und neuen BWINF-Teilnehmern zu fördern. Bei der Verlosung von zweien der Linux-Pakete nehmen diejenigen teil, die schon mal beim BWINF mitgemacht haben, beim 22. BWINF wieder dabei sind und eine Person ohne BWINF-Erfahrung für die Teilnahme am 22. BWINF werben konnten. Der oder die Geworbene muss dazu Namen und Geburtsdatum des/derwerbenden bei der Anmeldung angeben.

Raff

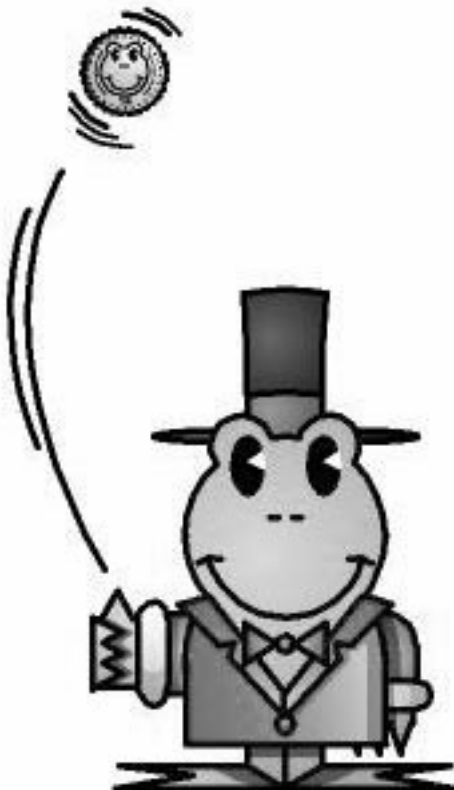
Der berühmte, geldgierige Dagobert beobachtet den Geldmarkt in der Stadt Valuta im Viel-Länder-Eck immer sehr genau. Jeden Donnerstag kann man dort an vielen einzelnen Ständen Geld von einer Währung in eine andere tauschen. Dagobert hat festgestellt, dass er bei den unterschiedlichen Tageskursen oft durch mehrere geschickte Tauschvorgänge bei verschiedenen Händlern sein Geld vermehren kann. Dagobert geht über den Markt und notiert sich Ankaufskurse in einer Tabelle. Heute hat er folgende Tabelle zusammengestellt:

| Währung | Kurs |
|----------------|-------------------|
| 1 Bärentaler | 2,70 Mausmark |
| 1 Entenpeseta | 0,75 Krötendollar |
| 1 Entenpeseta | 0,70 Froschkronen |
| 1 Entenpeseta | 1,80 Wolfspfunde |
| 1 Froschkrone | 1,10 Krötendollar |
| 1 Froschkrone | 1,10 Entenpeseten |
| 1 Krötendollar | 2,00 Wolfspfunde |
| 1 Krötendollar | 1,90 Wolfspfunde |
| 1 Krötendollar | 1,05 Froschkronen |
| 1 Mausmark | 1,30 Entenpeseten |
| 1 Mausmark | 0,90 Krötendollar |
| 1 Wolfspfund | 0,70 Entenpeseten |
| 1 Wolfspfund | 0,20 Bärentaler |
| 1 Wolfspfund | 0,50 Mausmark |

Er könnte z.B. pro Entenpeseta 0,75 Krötendollar bekommen, dann pro Krötendollar 2,0 Wolfspfunde und dann pro Wolfspfund 0,7 Entenpeseten. Mit solchem Tauschzyklus würde er aus je einer Entenpeseta $(0,75 * 2,0 * 0,7) = 1,05$ Entenpeseten machen, also einen Gewinn von $(5/3)\%$ pro Tausch erzielen. Dagobert sucht natürlich, von einer beliebigen Währung ausgehend, nach einem Tauschzyklus mit maximalem Gewinn pro Tausch.

→ Aufgabe

- Schreibe ein Programm, das die Tageskurstabelle ein liest, prüft, ob es Tauschzyklen gibt und gegebenenfalls einen mit maximalem Gewinn pro Tausch bestimmt.
- Dokumentiere die Programmresultate für drei verschiedene Tageskurstabellen. Eine soll die oben angegebene Tabelle sein.



Keuch

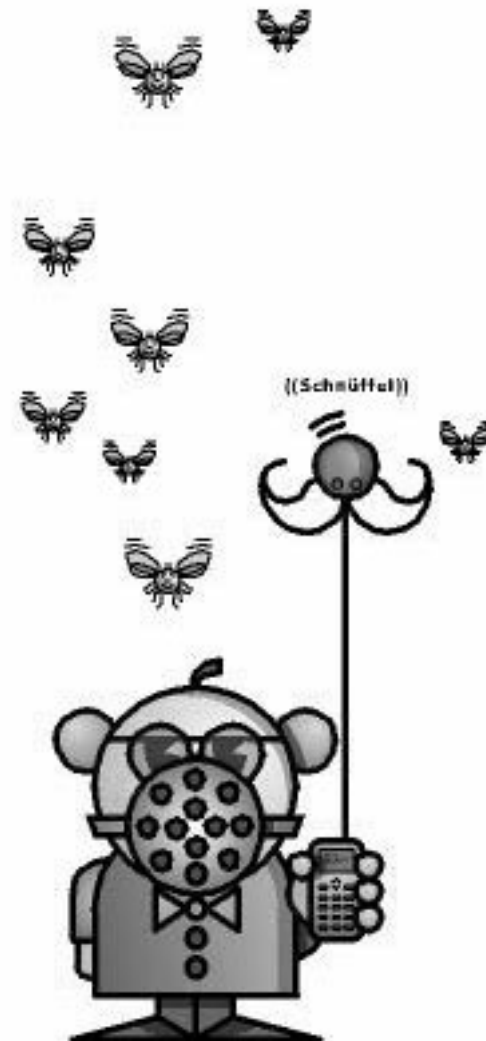
Die Bürgeraktion Sloterpeace will aktuelle Daten über die Atembarkeit der Luft frei verfügbar machen. Sie lässt ein besonderes Handy entwickeln, mit Mikrosensoren zur Messung der wichtigsten Stäube, Gase und Aerosole, welche die Luft verschmutzen oder vergiften können. Für jeden dieser Schadstoffe kennt das Handy einen Grenzwert. Die Grenzwerte lassen sich bei jedem Handy einzeln und beliebig einstellen, weil ja Asthmatiker, Raucher, Chemielaboranten usw. unterschiedliche Atembelastungen zu tolerieren bereit sind.

Bei möglichen Gefahren sollen die Handys ihre Besitzer warnen. Die folgende Alarmstrategie ist geplant:

- Wird bei einem Handy der Grenzwert für einen Schadstoff überschritten, alarmiert es akustisch und optisch seinen Besitzer.
- Gleichzeitig sendet das Handy an den Server seiner Funkzelle eine Gelb-Nachricht für diesen Schadstoff, die dieser an einige zufällig ausgewählte andere Handys in der gleichen Funkzelle weitersendet.
- Empfängt ein Handy gelegentlich einzelne Gelb-Nachrichten, tut es nichts.
- Empfängt ein Handy innerhalb kurzer Zeit etliche Gelb-Nachrichten bezüglich desselben Schadstoffes, alarmiert es seinen Besitzer, auch wenn sein entsprechender Grenzwert nicht überschritten ist.
- Empfängt der Zellenserver innerhalb kurzer Zeit eine größere Anzahl Gelb-Nachrichten bezüglich desselben Schadstoffes, sendet er eine entsprechende Rot-Nachricht an alle Handys in seiner Funkzelle und an alle Server der benachbarten Funkzellen.
- Empfängt ein Handy eine Rot-Nachricht, alarmiert es seinen Besitzer, auch wenn der entsprechende Grenzwert bei ihm nicht überschritten ist.

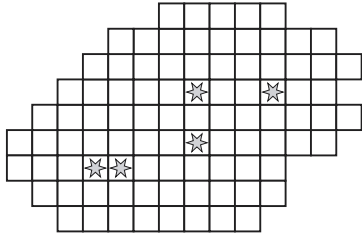
→ Aufgabe

- Liste alle Sprachformeln (z.B. „einige“) aus der Alarmstrategie auf, für die beim Programmieren ein numerischer Wert eingesetzt werden muss. Wieviele Strategievariablen ergeben sich daraus?
- Programmiere die Alarmstrategie eines Handys mit Mikrosensoren für 10 verschiedene Schadstoffe.
- Programmiere die Alarmstrategie eines Zellenservers für 100 in seiner Zelle aktive Handys und 10 benachbarte Zellen.
- Beschreibe in einem kurzen Text, wie Handybesitzer, eventuell gemeinsam, Alarme bewusst auslösen könnten. Kann man das mit einer modifizierten Alarmstrategie verhindern?
- Begründe in einem kurzen Text, welche Erweiterungen der Alarmstrategie des Zellenservers sinnvoll wären.



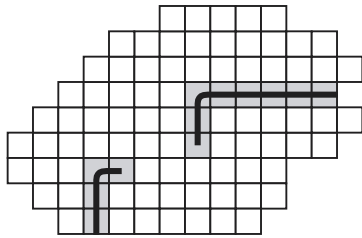
Grab

In einem Berg sollen Diamantenvorkommen erschlossen werden, die sich alle in einer einzigen geologischen Schicht befinden. Für diese Schicht liegt ein Prospektoren-Plan vor. Alle Planquadrate, in denen Diamantenvorkommen vermutet werden, sind darin gekennzeichnet. Ein Beispiel:



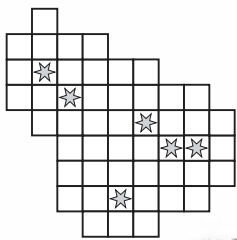
Um die Diamantenvorkommen zu erschließen, müssen entlang der geologischen Schicht von außen Stollen getrieben werden. Mit den verfügbaren Maschinen kann ein Stollen nur in nördlicher, östlicher, südlicher oder westlicher Richtung in das nächste Planquadrat geführt werden (also nicht schräg). Aus Sicherheitsgründen dürfen Stollen auch nicht verzweigen; von jedem Planquadrat aus geht es also höchstens in ein Vorgänger- und ein Nachfolger-Quadrat.

Die Grabungen sollen kostengünstig durchgeführt werden. Gesucht ist eine Erschließungslösung, zu allen Diamantenvorkommen zu gelangen und dabei - einschließlich der Fundorte selbst - in möglichst wenig Planquadraten zu graben. Im Beispiel sind das mindestens 12:



→ Aufgabe

1. Schreibe ein Programm, das einen Prospektoren-Plan einliest, eine mit minimalem Grabungsaufwand verbundene Erschließungslösung berechnet und die zu grabenden Stollen in geeigneter Weise ausgibt.
2. Demonstriere das Programm an mindestens drei Plan-Beispielen mit maximal 10 vermuteten Diamantenvorkommen und stelle jeweils die gefundene Lösung grafisch dar. Eines der Beispiele sei der folgende Plan:



Flitz

Goldfische haben ihre Eigenheiten. Jeder Fisch fühlt sich zu gewissen Fischen hingezogen und meidet andere. So entwickelt sich im Goldfischteich ein munteres Treiben.

→ Aufgabe

1. Modelliere einen Teich mit einigen Fischen und ihrem Sozialverhalten - ohne Einwirkung der Umgebung - und beschreibe Deine grundlegenden Entscheidungen für die Modellierung.
2. Visualisiere das muntere Treiben im Teich durch die Darstellung der Fische als Kreise in verschiedenen Farben.
3. Simuliere mit Deinem Modell die folgenden Gegebenheiten:
 - Ein Fisch wird von allen anderen geliebt, möchte aber mit keinem Fisch zusammentreffen. Das Verhalten der anderen Fische ist beliebig.
 - Jeder Fisch erhält zufällig Zu- bzw. Abneigungen zugeordnet, so dass das Verhältnis seiner Zu- und Abneigungen 70 zu 30 beträgt.
 - Jeder Fisch erhält zufällig Zu- bzw. Abneigungen zugeordnet, so dass das Verhältnis seiner Zu- und Abneigungen 30 zu 70 beträgt.

Babbel

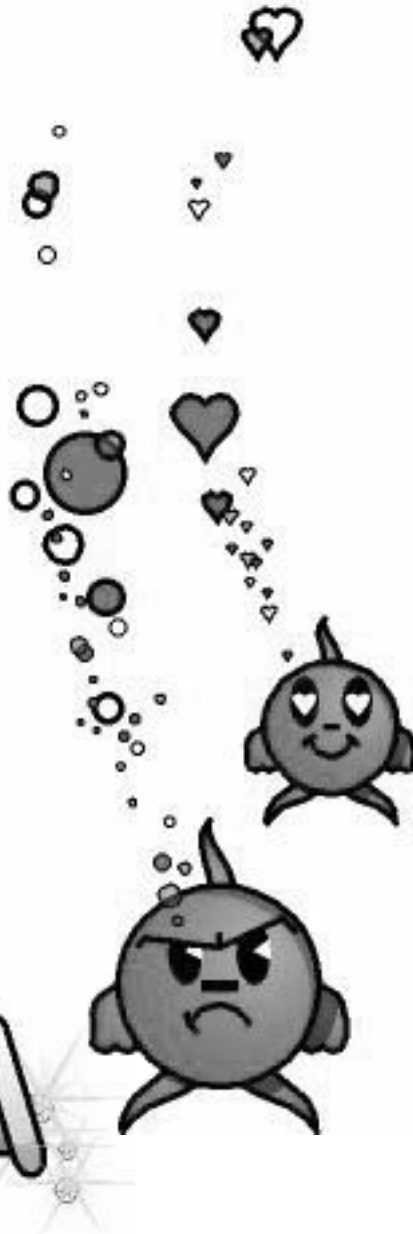
Der Schnellprogrammierer BaSiX möchte unbedingt den Leuten helfen, die Schwierigkeiten mit Übersetzungen vom Englischen ins Deutsche haben. In der nächsten Schulpause implementiert er dazu eine Wort-für-Wort-Übersetzung auf der Basis von Lexika, die so strukturiert sind wie das folgende Lexikon-Beispiel:

```
((cat Katze)(cats Katzen)(the das)(into in)(after nach)  
(eat fressen)(eats frisst)(small klein)(later später)  
(the die)(the der)(mouse Maus)(hole Loch)(mice Mäuse)  
(big groß)(run rennen)(runs rennt)(after hinterher))
```

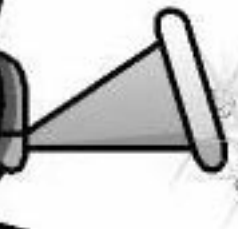
Irgendwie sind die Leute aber nicht zufrieden.

→ Aufgabe

1. Konstruiere mit den Vokabeln des Lexikon-Beispiels einige englische Sätze und ihre deutschen "Übersetzungen". Finde vor allem Sätze, bei denen die Schwächen einer Wort-für-Wort-Übersetzung deutlich werden. Charakterisiere die Schwächen.
2. Im Internet wird Funktionalität zur Sprachübersetzung angeboten, z.B. unter www.babelfish.org und www.google.com. Stelle die Leistungsfähigkeit dreier solcher Übersetzungsfunktionen auf die Probe. Finde möglichst kurze und einfache Sätze, die von mindestens einer Funktion unzureichend übersetzt werden. Stelle begründete Vermutungen über die Ursache der Fehler an.
3. Erläutere kurz drei der vielen fundamentalen Schwierigkeiten bei einer maschinellen Übersetzung natürlichsprachlicher Texte. Gib Beispiele an, welche diese Schwierigkeiten treffend deutlich machen.



```
((Shift) (Delicious) (Echt) (Tacker))
```



Allgemeine Hinweise

Zu den Aufgaben sende uns jeweils Folgendes:

→ **Lösungsidee:**

Eine Beschreibung der Lösungsidee, unabhängig vom eingesandten Programm. Die Idee und die zu ihrer Beschreibung verwendeten Begriffe müssen aber im Programm nachvollziehbar sein.



→ **Programm-Dokumentation:**

Eine Beschreibung, wie die Lösungsidee im Programm und seinen Teilen umgesetzt wurde. Hinweise auf Besonderheiten und Nutzungsgrenzen, verlangte Eingabeformate usw.

→ **Programm-Ablaufprotokoll:**

Kommentierte Probeläufe des Programms, d.h. wann wird welche Eingabe erwartet, was passiert dann, welche Ausgabe erscheint usw. Mehrere unterschiedliche Beispiele, die die Lösung der Aufgabe und das Funktionieren des Programms verdeutlichen (teilweise in den Aufgabenstellungen vorgegeben). Bildschirm-Fotos sind zulässig.

→ **Programm-Text:**

Den kommentierten Quelltext des Programms in einer der gängigen höheren Programmiersprachen wie z. B. Pascal. Skriptsprachen sind zulässig, Maschinsprache nicht. Den Programmtext bitte ausdrucken, dabei aber auf nicht selbst geschriebene Teile (wie z. B. Funktionen der Entwicklungsumgebung und automatisch generierten Programmtext) verzichten.

→ **Programm:**

Das lauffähige Programm selbst auf einer CD oder 3,5"-Diskette, die unter DOS bzw. Windows98 gelesen werden kann. Die CD/Diskette muss auch den Programm-Text enthalten.

Ist kein Programm gefordert, strukturiere deine Aufgabenbearbeitung der Aufgabenstellung entsprechend.

→ **Einsendungen werden danach bewertet,**

- ob die Aufgaben vollständig und richtig bearbeitet wurden,
- ob die Ausarbeitungen gut strukturiert und verständlich sind und
- ob die (Programm-) Unterlagen vollständig, übersichtlich und lesbar sind.

Bitte schicke deine Arbeit nach Aufgaben geordnet und geheftet auf einseitig bedrucktem DIN-A4-Papier. Nummeriere alle Blätter rechts oben und versieh sie mit deinem Namen. Die Texte sollen in Deutsch abgefasst sein. Falls du DIN-A4-Klarsichthüllen mit Lochrand zur Hand hast, stecke bitte jeweils alles, was zu einer Aufgabe gehört, in eine solche Hülle. Andernfalls loche die Blätter bitte.

Beschrifte die CD oder Diskette, die die Programme und Programm-Texte enthält, mit deinem Namen.

Fülle das Anmeldeformular (Klappe des Aufgabenblattes oder eine Kopie davon) vollständig, korrekt und leserlich! aus. Insbesondere das Geburtsdatum muss angegeben sein, denn sonst kann die Einsendung nicht bewertet werden. Teilnehmer, die die Schule bereits verlassen haben, geben bei „Klassenstufe“ bitte an, was sie zur Zeit machen. Erstteilnehmer nennen ggf. den Teilnehmer, der sie zum Mitmachen angeregt hat, mit Namen und Geburtsdatum.

Bei Gruppen muss jeder Teilnehmer ein Formular ausfüllen, Gruppenmitglieder ohne Anmeldeformular werden nicht gewertet!

Für die Anmeldung gibt es unter www.bwinf.de auch ein Internet-Formular, das vor dem Einsendeschluss ausgefüllt werden kann. Bei dieser Online-Anmeldung wird eine Kennnummer vergeben; bei der Einsendung muss das Papierformular nur noch mit dieser Nummer, dem Namen und einer Unterschrift versehen werden. Wer sich per Internet anmeldet, erhält nach der Einsendung eine Eingangsbestätigung per E-mail.

Einsendungen per E-mail oder nur auf CD/Diskette ohne Ausdruck können wir leider nicht akzeptieren. Auch alle geforderten Beispiele müssen auf Papier dokumentiert sein. Es ist nicht auszuschließen, dass die Gutachter nur auf die Papierunterlagen zugreifen können.

→ **Sende alles an:**

Bundeswettbewerb Informatik
Ahrstraße 45
53175 Bonn



→ **Für Fragen zu den Aufgaben gibt es eine Hotline:**

Telefonisch unter 0228 / 37 86 46 jeweils von 17-19 Uhr am 22.9., 14.10. und 12.11. 2003 oder E-mail an: bwinf@bwinf.de



oder schriftlich an die obige Adresse bzw. per Fax an 0228 / 37 29 000.

Informationen (allgemeine Tipps und FAQ) gibt es auch im Internet unter <http://www.bwinf.de>.

Meinungen und Fragen (aber keine Lösungsideen) zu den Aufgaben können auch in der BWINF-Newsgroup fido.ger.bwinf ausgetauscht werden.

→ **Einsendeschluss ist der 17. November 2003** (Datum des Poststempels).

Verspätete Einsendungen können nicht berücksichtigt werden. Der Rechtsweg ist ausgeschlossen. Die Einsendungen werden nicht zurückgegeben. Der Veranstalter erhält das Recht, die Beiträge in geeigneter Form zu veröffentlichen.

Wer wissen möchte, ob seine Einsendung angekommen ist, kann eine an sich selbst adressierte und frankierte Postkarte mitschicken oder sich auf den BWINF-Webseiten mit E-mail-Adresse anmelden.



Anmeldung

22. Bundeswettbewerb Informatik 2003 / 2004, 1. Runde

Bitte in Druckschrift ausfüllen und Zutreffendes ankreuzen.



Geburtsdatum Name, Vorname

Straße männlich weiblich

Postleitzahl Wohnort

Telefon: Vorwahl / Durchwahl E-mail-Adresse

Name der (ehemaligen) Schule Schultyp

Informatiklehrer/in Klassenstufe (1-13)

Straße der Schule

Postleitzahl Schulort Bundesland der Schule

Namen anderer Gruppenmitglieder

Wieviele Stunden hast du gebraucht?

Was hast du bei der Bearbeitung verwendet?
Welche/s/n ...

Programmiersprache? Betriebssystem? Computer(typ)? eigener

Hast du an anderen Wettbewerben teilgenommen?
Informatik Mathematik sonstige

Wie hast du vom Wettbewerb erfahren?
schon mal teilgenommen Schule/Lehrer ehemalige Teilnehmer sonstiges

Geworben durch ehe- und diesmaligen Teilnehmer

Name, Vorname, Geburtsdatum

→ Diese Daten werden an niemanden weitergegeben, haben keinen Einfluss auf die Bewertung, aber dienen statistischen Zwecken. Sie werden ausschließlich für die Zwecke des Bundeswettbewerbs Informatik ausgewertet.

→ Ich bin mit der Computerspeicherung dieser Daten einverstanden und versichere, dass ich die Aufgaben selbstständig bzw. mit den angegebenen Gruppenmitgliedern bearbeitet habe.

Datum Unterschrift



Beispielaufgabe: Graf Rüdiger

Der schrullige Graf Rüdiger ist bekannt für seine spleenigen Ideen, mit denen er die Besucher auf seiner Burg in Lüne immer wieder überrascht. Jetzt hat er sich für seine Eingangspforten verspielt, aufwändige Schließanlagen konstruiert lassen. Eine Schließanlage besteht aus N Riegeln, deren Stellungen (auf oder zu) von außen sichtbar sind. Sie können einzeln nur nach den folgenden Regeln bewegt werden:

- Riegel 1 kann immer bewegt werden.
- Riegel 2 kann nur bewegt werden, wenn Riegel 1 auf ist.
- Jeder andere Riegel kann genau dann bewegt werden, wenn der Riegel mit der nächst kleineren Nummer auf ist und alle mit noch kleineren Nummern zu sind.
- Der Riegel N kann außerdem noch bewegt werden, wenn alle Riegel 1 bis N-1 zu sind.

Eine Pforte lässt sich nur öffnen, wenn alle Riegel auf sind.

Beispiel für N=4

(in der letzten Zeile der Tabelle bedeutet Ri, dass Riegel i bewegt wurde):

| | Pforte (ordentlich) | | | | | | | | | | | | Pforte | |
|----------|---------------------|---|---|---|---|---|---|---|---|---|---|---|--------|---|
| | zu | | | | | | | | | | | | auf | |
| Riegel 1 | z | a | a | z | z | a | a | z | z | a | a | z | a | a |
| Riegel 2 | z | z | a | a | a | a | z | z | z | z | z | z | z | a |
| Riegel 3 | z | z | z | z | a | a | a | a | a | a | a | a | a | a |
| Riegel 4 | z | z | z | z | z | z | z | z | z | a | a | a | a | a |
| z = zu | | | | | | | | | | | | | | |
| a = auf | | | | | | | | | | | | | | |
| | | | | | | | | | | | | | | |

Graf Rüdiger hat mehrere Schließanlagen mit unterschiedlich vielen Riegeln bauen lassen. Leider kommt es häufig vor, dass Besucher einige Riegel bewegen, die Pforte aber nicht öffnen können und die Schließanlage unordentlich hinterlassen. Für Graf Rüdiger ist eine Pforte nur dann ordentlich geschlossen, wenn alle Riegel zu sind. Jeden Abend muss deshalb der Diener von Graf Rüdiger durch die Burg gehen und an allen Pforten die Riegel schließen. Du sollst ihm helfen.

→ Aufgabe

Schreibe ein Programm, das für ein beliebiges N und einen beliebigen Zwischenzustand einer Schließanlage die Folge derjenigen Riegel ausgibt, die der Diener bewegen muss, um die Pforte ordentlich zu schließen. Erzeuge für drei verschiedene Schließanlagen mit N > 4 und je zwei Zwischenzustände solche Bewegungsfolgen. Eine Bewegungsfolge davon soll für N = 6 und den Zwischenzustand (zu,zu,zu,auf,auf) erzeugt werden.

→ Lösungsidee

Beim Betrachten der vier Regeln zum Verschieben der Riegel fallen zwei Dinge auf:

- Regel 4 ist überflüssig. Sie dient lediglich dem Bewegen von Riegel N, der aber bereits mit Regel 3 (für N>2) bzw. Regel 2 (N=2) oder Regel 1 (N=1) bewegt werden kann. Regel 4 ermöglicht unter bestimmten Voraussetzungen ein „Abkürzen“ – da aber in der Aufgabenstellung nicht nach der kürzesten Schließfolge gefragt war, kann man auf das Anwenden dieser Regel verzichten. Dadurch vereinfacht sich das Programm, da es nun nur noch drei Regeln zu beachten gibt.

- Um einen beliebigen Riegel x bewegen zu können, ist lediglich die Stellung der Riegel mit Nummern kleiner x von Bedeutung. Diese müssen in eine bestimmte Stellung entsprechend der Regeln 1 bis 3 gebracht werden, bevor Riegel x bewegt werden kann. Andererseits beeinflusst die Stellung eines Riegels die Beweglichkeit der Riegel mit niedrigerer Nummer nicht. Es müssen also zuerst die Riegel mit hohen Nummern in die Endstellung gebracht werden, bevor dann die Riegel mit kleineren Nummern in ihre Zielstellung gebracht werden können.

Um die Pforte ordentlich zu schließen, beginnt man also mit Riegel N. Wenn man Glück hat, so ist Riegel N bereits geschlossen. Hat man Pech und Riegel N ist geöffnet, so müssen zuerst die Riegel N-1 bis 1 in die benötigte Stellung entsprechend Regel 3 bzw. 2 gebracht werden, bevor Riegel N endgültig geschlossen werden kann. Auf die gleiche Weise schließt man danach der Reihe nach Riegel N-1 bis Riegel 1.

Es bleibt nun noch zu klären, wie man die Riegel mit den kleineren Nummern in die benötigten „Zwischenpositionen“ bekommt. Dazu kann man auch das soeben beschriebene (rekursive) Verfahren verwenden, das von der zu erzielenden Stellung und der Menge der zu bearbeitenden Riegel unabhängig ist. Das folgende Pseudoprogramm formalisiert diese Verallgemeinerung:

```

Bringe m Riegel in Stellung s_m, ..., s_1:
  Wenn Riegel m nicht in Stellung s_m dann
    t_(m-1), ..., t_1 sei Stellung um Riegel m
    bewegen zu können
    Bringe m-1 Riegel in Stellung t_(m-1), ..., t_1
    Bewege Riegel m
  end
  Bringe m-1 Riegel in Stellung s_(m-1), ..., s_1
end
    
```

Um Graf Rüdigers Diener mit dieser Prozedur zu helfen, muss dann nur noch

```
Bringe N Riegel in Stellung zu, ..., zu
```

gerufen werden.

→ Programm-Dokumentation

Das Lösungsprogramm ist in Prolog verfasst, da sich das Problem sehr gut logisch formulieren lässt.

Der Zustand des Schlosses wird als Liste von Werten aus der Menge {auf, zu} dargestellt. Dabei steht der Zustand des Riegels N an erster Stelle in der Liste, der Zustand des Riegels N-1 an zweiter Stelle und so weiter. Der Zustand des Schlosses aus dem Pflichtbeispiel der Aufgabenstellung sieht zum Beispiel so aus:

```
[zu, auf, auf, zu, zu, zu]
```

Der Zustand „ordentlich geschlossen“ sieht entsprechend so aus:

```
[zu, zu, zu, zu, zu, zu]
```

Die Folge der bewegten Riegel wird ebenfalls als Liste repräsentiert. Dabei steht an erster Stelle der Liste die Nummer des Riegels, der als erster bewegt wurde, an zweiter Stelle der Riegel, der als zweiter bewegt wurde, und so weiter. Die im Aufgabenblatt durchgeführten Bewegungen sehen zum Beispiel so aus:

```
[1, 2, 1, 3, 1, 2, 1, 4, 1, 2]
```

Zur Lösung des Problems wird ein Prädikat `graf(Ist, Bewegungen, Soll)` definiert. Dieses Prädikat ist erfüllt, wenn man durch Bewegen der in Bewegungen festgelegten Riegel entsprechend obiger Strategie vom Ist- in den Soll-Zustand gelangt. Lässt man bei einer Anfrage mit diesem Prädikat den zweiten Parameter variabel, wird das Prolog-Programm diese Variable mit einer Bewegungsfolge belegen.

Um die Beispielaufgabe zu lösen, muss also folgende Anfrage an das Prolog-System gerichtet werden:

```
graf([zu, auf, auf, zu, zu, zu], Bewegungen,
[zu, zu, zu, zu, zu, zu]).
```

→ Programm-Ablaufprotokolle

Nach Starten des Interpreters wird zuerst der Quellcode eingeladen.

```
?- consult('bwinf21.1').
% bwinf21.1 compiled 0.01 sec, 2,324 bytes
```

Nun kann das in der Aufgabenstellung geforderte Beispiel gelöst werden.

```
?- graf([zu, auf, auf, zu, zu, zu], Bewegungen,
[zu, zu, zu, zu, zu, zu]).
Bewegungen = [5, 1, 2, 1, 3, 1, 2, 1, 4, 1, 2,
1, 3, 1, 2, 1]
Yes
```

Belegt man alle Parameter, kann man zum Beispiel herausfinden, ob das in der Aufgabenstellung genannte Beispiel tatsächlich korrekt ist.

```
?- graf([zu, zu, zu, zu], [1, 2, 1, 3, 1, 2, 1,
4, 1, 2], [auf, auf, auf, auf]).
Yes
```

Keht man probierhalber die Reihenfolge der Schließvorgänge um, so ergibt sich keine gültige Lösung.

```
?- graf([zu, zu, zu, zu], [2, 1, 4, 1, 2, 1,
3, 1, 2, 1], [auf, auf, auf, auf]).
No
```

Belegt man nun alle Parameter bis auf den letzten, kann man sich zu einem gegebenen Anfangszustand und einer gegebenen Bewegungsfolge den erreichten Endzustand der Schließanlage anzeigen lassen.

```
?- graf([zu, zu, zu, zu], [1, 2, 1, 3, 1, 2, 1, 4],
Schloss).
Schloss = [auf, auf, zu, zu]
Yes
```

Ist statt dessen nur der Endzustand und die Schließfolge bekannt, kann man den Anfangszustand der Schließanlage bestimmen.

```
?- graf(Schloss, [1, 2, 1, 3, 1, 2, 1, 4],
[auf, auf, zu, zu]).
Schloss = [zu, zu, zu, zu]
Yes
```

→ Programm-Text

Zunächst einige Worte zu Prolog-Programmen: In diesen beschreibt man Prädikate (quasi Funktionen, die nur die Wahrheitswerte wahr und falsch liefern können) und deren Erfüllbarkeit (wann der Wert wahr geliefert wird). Eine solche Beschreibung kann aus mehreren Fakten (wie im folgenden Programm bei `ok` und `falsch`, deren Gültigkeit sich allein aus festen Parameterwerten ergibt) oder Regeln (wie bei `graf`, wo hinter dem `:-` Vorbedingungen für die Erfüllbarkeit angegeben sind) bestehen. Die Notation `[First|Rest]` beschreibt eine Liste mit dem ersten Element `First` und der Liste der weiteren Elemente `Rest`. Bezeichner, die mit einem Großbuchstaben beginnen, sind Variablen, deren Belegung interessiert, während der Unterstrich `_` eine beliebige Variable ist. Hat eine Prädikatbeschreibung mehrere Elemente, werden diese bei der Beantwortung einer Anfrage (die Überprüfung einer Regelbedingung ist wieder eine Anfrage) von oben abgearbeitet. Enthält eine Anfrage Variablen, werden automatisch deren mögliche Belegungen durchprobiert – mit dem Ziel, eine Belegung zu finden, mit der die Anfrage erfüllt wird. Nun aber das Programm:

```
% Welche Kombination aus Ist- und Soll-Zustand
% ist ok...
ok(auf, auf).
ok(zu, zu).
```

```
% ...und welche Kombination ist falsch.
falsch(auf, zu).
falsch(zu, auf).
```

```
% Wenn es keine Riegel gibt, braucht man auch keine
% zu bewegen.
graf([], [], []).
```

```
% Ist der oberste Riegel ok, so brauchen wir uns nur
% noch um die restlichen Riegel zu kümmern.
graf([X|Xs], S, [Y|Ys]) :-
  ok(X, Y),
  graf(Xs, S, Ys).
```

```
% Ist der oberste Riegel in der falschen Stellung,
% so müssen wir zuerst die kleineren Riegel ent-
% sprechend der Vorbedingungen der Regeln 1-3 ein-
% stellen, dann den obersten Riegel bewegen (also
% seine Nummer in die Liste mit den Bewegungen
% aufnehmen) und dann die restlichen Riegel in die
% endgültige Position bewegen.
graf([X|Xs], S, [Y|Ys]) :-
  falsch(X, Y),
  vorbedingung(Xs, Zs),
  graf(Xs, L1, Zs),
  length([X|Xs], N),
  append(L1, [_|_L2], S),
  graf(Zs, L2, Ys).
```

```
% Für den ersten Riegel gibt es keine Vorbedingung.
vorbedingung([], []).
```

```
% Ab dem zweiten Riegel muß der nächstkleinere Riegel
% offen sein, und der Rest (wenn es noch kleinere
% Riegel gibt) geschlossen.
vorbedingung([_|_Ls], [auf|Xs]) :-
  restzu(Ls, Xs).
```

```
% Prüfe, ob in einer Liste von Riegeln alle zu sind.
restzu([], []).
restzu([_|_Ls], [zu|Xs]) :-
  restzu(Ls, Xs).
```

```
% Prüfe, ob in einer Liste von Riegeln alle zu sind.
restzu([], []).
restzu([_|_Ls], [zu|Xs]) :-
  restzu(Ls, Xs).
```

```
restzu([_|_Ls], [zu|Xs]) :-
  restzu(Ls, Xs).
```

Lösung von Michael Schneider