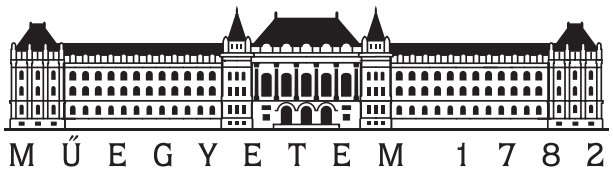

ROYAL INSTITUTE
OF TECHNOLOGY

6th BME-KTH International

24-hour Programming Contest

Problem Set

April 21-23, 2006
Budapest, Hungary

http://www.challenge24.org

The main sponsor of the event is Fornax Co.

Platinum Grade Sponsor

GRAPHISOFT®
Virtual Building Solutions

International Golden Grade Sponsor

National Golden Grade Sponsors

Silver Grade Sponsors

John von Neumann Computer Society, IEEE, HTE
Professional Sponsors

CONTENTS

Introduction

Welcome to the 6th International 24-hour Programming Contest!

This year's contest features seven problems of very varying nature. They do have a common denominator, though, namely that they are all centered around The Daily Task, a large (though largely unknown), international newspaper. As spreading information is the primary purpose of a newspaper, and fast publication of the latest news one of the celebrated virtues in the media industry, the 6th eXtreme challenge also features, for the first time, live judging and continuous reporting of the standings.

No programming contest would be complete without some classic problem solving, and that's what you will face in the first chapter, when looking behind the scenes of newspaper production.

The Daily Task is looking to expand internationally, and in Chapter 2 you will help them by devising some audiovisual communication solutions.

Chapter 3 features some relaxation, as all you will need to do is play a game. Or, well, make your computers play the game – against the other teams.

Chapter 4 contains problems of a more secretive nature. If you manage to figure out what it's about, you've come a long way...

Like most newspapers, The Daily Task has a sports section. Currently, they are reporting from the Budapest Slot Car Racing Grand Prix, which is described in Chapter 5 and in which *you* will compete, by writing programs that control actual cars on an actual race track.

In Chapter 6, you will help the owners of The Daily Task market themselves by producing a promotional video and by participating in a PR event.

Finally, in Chapter 7 everything is tied together in a problem where you will need to follow the contest standings and use that information to your advantage. In this chapter, you will be making the economy section of The Daily Task more interesting by manipulating the stock market.

Remember: you only have 24 hours, so choose wisely which tasks to attack!

Good luck and have fun!

Scoring Summary

Below you will find the maximum score for each chapter as well as the total amount of points available in the contest.

Behind the Scenes	**1,200 points**
Communication in Bordomia	**1,000 points**
Strategic Thinking	**1,200 points**
Cryptokuo Chawaelleyiengekee	**1,000 points**
The Budapest Grand Prix	**1,200 points**
Promotional Activities	**900 points**
Manipulating the Stock Market	**1,000 points**
Grand Total	**7,500 points**

General Remarks

Some general comments about the tasks:

- All forms of wireless networking (WiFi, Bluetooth, IrDA etc.) are **strictly forbidden**. Violation of this rule can result in deduction of points or even disqualification from the contest. Naturally, the same is true for mobile phones and other means of communicating with the outside world.
- You can download sample files and useful documents, and view the live scoreboard and the stock market (see chapter 7) at `http://www.c24.finals/`.
- Unless otherwise stated, the newline character in text files is a single line feed character (decimal code 10). We will give you text files in this format and we expect that your output files also follow this convention.
- Only integer points will be awarded. If scoring of a problem results in a non-integer number of points, the value will be rounded to an integer value.
- Contact with the judges during the contest will be handled via e-mail. You have a POP3 e-mail account on the server `mail.c24.finals` (which should also be used as SMTP server). You can contact the judges at `judges@c24.finals`, and you should specify `Tn@c24.finals`, where `n` is your team number as seen on the next page, as your own address. Please check mail on this address regularly, as we may send important announcements.
- You will be given a separate document with username and password information for your e-mail account, web-based submission and all tasks requiring login to some sort of server.

Participating Teams

The following teams made it to the finals.

ID	Name	Country
T1	Garbage Collectors	Poland
T2	qwertY	Poland
T3	Trial Version	Hungary
T4	UPC Barcelona Tamarros	Spain
T5	Lag Bernhardsson	Sweden
T6	Teamisoara	Romania
T7	miners	Poland
T8	Végzetes Kivétel	Hungary
T9	pizza2code	Hungary
T10	Fast and Furious Most Wanted	Hungary
T11	Rusty	Poland
T12	Script Kiddies	Hungary
T13	Buheratorz	Hungary
T14	AEIOU	Austria
T15	Blue Screen Development	Hungary
T16	Royal Air Force	Serbia and Montenegro
T17	beer-to-beer	Hungary
T18	puddingforce	Hungary
T19	UPC-6	Spain
T20	The Breakpoints	Poland
T21	Ragacs	Hungary
T22	Les Blotières	Sweden
T23	PTrolls	Hungary
T24	Garfield Beheaders	Spain
T25	Jagiellonian North-South	Poland
T26	Lions Nis	Serbia and Montenegro
T27	UPC - Si es que va!	Spain
T28	Texas Codeboys	USA
T29	MacMix	Hungary
T30	Mit nekem te zordon Kárpátoknak fenyvesekkel vadregényes tája	Hungary

CHAPTER 1

Behind the Scenes

Life at a newspaper is more than just reporting, there are lots of other tasks to be solved. In this problem, you will get to help the editor-in-chief, Eddie Taur, by solving a few of the tasks that are part of the everyday operation of The Daily Task.

There are six tasks to be solved. For each task, you are given 10 input files. Each solved input file can give as many as 20 points.

The input files are available at contest website mentioned in the general remarks. You should produce solutions to the input files and submit them to the judges, also via the web page. When solved correctly, your solution will receive a score based on the quality of your solution, the time of your submission, and the total number of attempts for this test file so far. This score is rounded to the nearest integer. The total score for a task is computed by adding up the rounded scores of the best submissions for the individual test files.

The base score, including how the quality of the solution is measured, is described individually for each problem. The *total time* of a submission is the number of hours passed since the beginning of a contest, plus the number of submissions *prior* to this submission (i.e. 0 for the first submission). In other words, each extra submission costs one hour. The penalty incurred by time and submission count increases exponentially, and is given by $0.25 \cdot 0.75^{24-T}$, where T is the total time of the submission. For convenience, we have plotted the penalty function in Figure 1.1 and given the penalty values for some values of T in Table 1.1.

T	Penalty
6	4%
12	7%
18	13%
24	25%
30	47%

Table 1.1: The penalty function for some chosen values of T

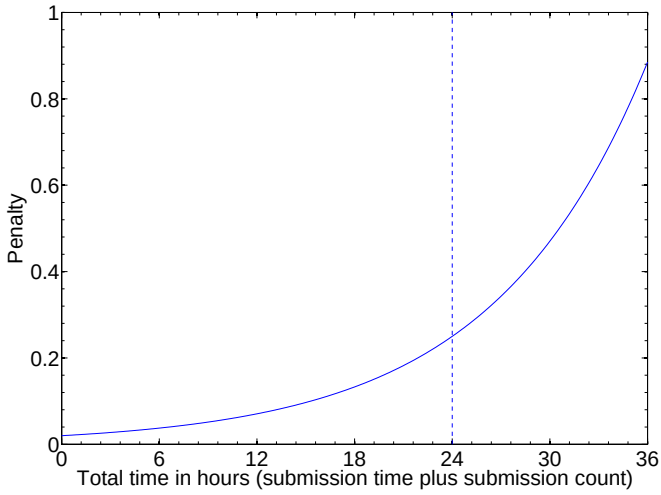

Figure 1.1: The penalty function

1.1 The Currency Exchange Affair

N. Cockbull is an up-and-coming reporter shooting for the stars, and perhaps a Pulitzer or two. For his next story, he is investigating the currency exchange market. He has discovered that through a careful sequence of currency exchanges, it is possible to earn new money.

For instance, let's say that the dollar-to-euro exchange rate was 123 dollars per 100 euro, and that the euro-to-dollar exchange rate was 82 euro per 100 dollars. You could then exchange €100 to $123, and then exchange the $123 to €100.86, having earned 86 eurocents!

Eddie Taur, being healthily skeptic, suspects that Cockbull might be exaggerating just a bit, so he wants to double-check the story. He has landed you with the task of finding out whether it really is possible to make money by currency exchange.

Input

The input consists of several test files, each containing several sets of exchange rates to be checked. Each input file starts with one integer T ($1 \leq T \leq 50$) giving the number of test sets in this file. Each test set starts with one integer N ($1 \leq N \leq 1000$) giving the total number of different currencies. Then follow N lines, each containing N real numbers with at most two decimals. The j:th number of the i:th line contains the amount of the j:th currency that you will get when exchanging 100 of the i:th currency.

Output

The output consists of a single line, containing either the message "Front page!" if Cockbull's story is true, or "Check your sources." if Cockbull is wrong.

Example test file

The following is a small example of an input file.

```
2
4
100.0 101.0 101.0 101.0
99.0 100.0 101.0 101.0
98.0 99.0 100.0 101.0
98.0 98.0 99.0 100.0
2
100.0 200.0
50.0 100.0
```

The output for this file should be

```
Front page!
Check your sources.
```

Scoring

The score of a submission is 20 if the output is correct, minus the penalty incurred by time and submission count. If the output is incorrect, it will receive 0 points.

Per test file (there are 10 test files):	**up to 20 points**

1.2 News Delivery Part 1: The Early Days

What good is running a newspaper if nobody reads it? Not much, so in order to really hit the market, Eddie Taur needs to set up delivery of the paper to the stores where it is sold. Initially, we want to make the paper available in N major cities. In each city we hire local distributors to take care of business there, thus reducing our job to transporting sufficiently many copies to each city. Sounds simple enough, doesn't it?

Well, it is! The Daily Task has a single printer which we assume is able to produce an infinite supply of papers. At their disposal is also a single car with a limited storage capacity. Every city has a certain demand of papers, and due to various bureaucratic policies of the local distributors, the entire demand needs to be supplied at a single time. In other words, we cannot deliver half of the papers needed at one time and then deliver the other half at a later time. Given the roads connecting the cities, we would like to find out the minimum distance the car needs to drive in order to deliver the news to all N cities.

Input

The input consists of several test files. Each test file consists of a test case to be checked. Each test file begins three integers, the number of cities N ($1 \leq N \leq 15$), the number of roads M ($0 \leq M \leq 200$), and the capacity of the truck, C ($1 \leq C \leq 200$). Then follow M lines, each containing three integers $u\,v\,d$ (where $0 \leq u, v \leq N$, $u \neq v$ and $1 \leq d \leq 20000$) indicating that there is a road of length d between cities u and v. The cities are numbered from 1 to N, and the printer is "city" 0. Then follow N lines, the i:th of which contains an integer d_i, the paper demand in city i ($1 \leq d_i \leq C$).

Output

The output should consist of a sequence of lines containing instructions for the truck driver, each instruction being one of

- `GO` v – Go to city v. This command requires that there is a road from the current city to city v.
- `DELIVER` d – Deliver d papers to the city we are currently in. d must be equal to the demand of newspaper in this city. This operation requires that the truck contains at least d papers.
- `DONE` T – This is the last command, indicating the end of the instructions. T should be the total distance travelled by the truck.

The truck always starts at the printer, and the truck should always return to the printer after having delivered all the papers (in other words, the last instruction before `DONE` T should be `GO 0`). Whenever the printer is visited, the truck is automatically refilled to the maximum capacity. The instructions should contain exactly N `DELIVER` commands, one in each city. The driving instructions may contain at most 200 commands.

Example test file

The following is a small example of an input file.

```
3 3 10
0 1 31
1 2 37
1 3 43
10
5
5
```

A possible solution to this input is

```
GO 1
DELIVER 10
GO 0
GO 1
GO 2
DELIVER 5
GO 1
GO 3
DELIVER 5
GO 1
GO 0
DONE 284
```

Scoring

If your output is valid, its score is $20 \cdot 0.5^{\frac{Val-Opt}{Opt}}$ minus penalty, where Val is the value of your solution, and Opt is the value of the best solution found so far by any team. If your output violates the rules, its score will be 0.

Per test file (there are 10 test files):	**Up to 20 points**

1.3 Office Wall Decoration

One of the most important aspects of being a succesful newspaper editor is having an elegantly decorated office wall, showing off diplomas, awards, and award-winning articles in the best possible way.

In this problem, you will help Eddie Taur decorate his office. Eddie has N decorations that he wants to hang on his wall. Furthermore, he has identified $M \geq N$ potential positions on his wall where the decorations can be placed.

So far so good. What makes things complicated is that, while some of the decorations go well together (for instance, putting Eddie's picture next to some diplomas), some of the decorations simply should not be mixed (for instance, putting Eddie's picture next to that award-winning article on the worst disaster to hit the world in the last 500 years). Thus, each pair of decorations i and j has a suitability score $v_{i,j}$ (which may be negative). When placing them at positions q_i and q_j, these two decorations contribute $\frac{v_{i,j}}{\text{dist}(q_i,q_j)^2}$ to the total value of this placement of wall decorations, where $\text{dist}(q_i, q_j)$ is the distance between the two positions.

The total value of a placement of the decorations is then

$$\sum_{i=1}^{N} \sum_{j=i+1}^{N} \frac{v_{i,j}}{\text{dist}(q_i, q_j)^2},$$

where q_i is the position where decoration i is placed. Your job is to find a placement of the decorations which maximizes this value.

Input

Each input file consists of one wall to be decorated. The description of this wall begins with the integers N and M, both at most 100, and an integer P, denoting the number of pairs with a non-zero suitability score. Then follow P lines containing three integers $i\, j\, v_{i,j}$, giving the suitability score of the decorations i and j. Each pair i, j will be given at most once, and any pair that is not present in the input has a suitability score of 0. $v_{i,j}$ has an absolute value of at most 10000.

Finally, there will be M lines, each containing two integers x and y giving a position where it is possible to hang a decoration. The wall is a two-dimensional plane with lower-left corner $(0, 0)$ and upper-right corner $(10000, 5000)$. The distance between two points (x_1, y_1) and (x_2, y_2) is the Euclidian distance $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. We ignore the size of the different decorations, and treat them as points.

Output

The output consists of N integers, $p_1, \ldots, p_N$, indicating that the i:th decoration should be placed at the p_i:th point. All N positions should be different and in the range 1 to M.

Example test file

The following is a small example of an input file.

```
3 4 2
1 2 100
1 3 -500
1 1
1 4
```

```
9 1
9 4
```

In this case, the most important thing is to keep decorations 1 and 3 away from each other, and then to keep 1 and 2 fairly close. So one good solution would be to put decoration 1 at $(9, 4)$, decoration 2 at $(9, 1)$, and decoration 3 at $(1, 1)$. The output corresponding to this solution is

```
4
3
1
```

The value of this placement is $\frac{100}{3^2} + \frac{-500}{8^2+3^2} \approx 4.262$. No better placement is possible for this test case.

Scoring

If your output is valid, its score is $20 \cdot 0.5^{\frac{Opt-Val}{100}}$ minus penalty, where Val is the value of your solution, and Opt is the value of the best solution found so far by any team. If your output violates the rules, its score will be 0.

Per test file (there are 10 test files):	**Up to 20 points**

1.4 News Delivery Part 2: A Growing Business

After a few years, the small scale news delivery operation set up previously is becoming insufficient, and it is time to scale things up a notch.

The setup is basically the same as in section 1.2. There are, however, two differences. First, due to The Daily Task's increasing popularity, there are now several more cities in which the paper is sold. Second, in order to be able to handle this, there are now several (say, P) printers, each with their own transport car (all having the same limited capacity). Each of the P cars can drive independently of the others. Furthermore, the cars can only refill their supply of papers at their own printer, not at any of the other printers.

The goal this time is to minimize the longest total distance driven by any of the cars.

Input

The input consists of several test files. Each test file consists of a test case to be checked. Each test file begins with four integers, the number of cities N ($1 \leq N \leq 50$), the number of printers P ($1 \leq P \leq 10$) the number of roads M ($0 \leq M \leq 1000$), and the capacity of the trucks, C ($1 \leq C \leq 200$). Then follow M lines, each containing three integers $u\, v\, d$ (where $0 \leq u, v \leq N$, $u \neq v$, and $1 \leq d \leq 20000$) indicating that there is a road of length d between cities u and v. The cities are numbered from 1 to N, and the printers are the "cities" with numbers $N+1$ to $N+P$. Then follow N lines, the i:th of which contains an integer d_i, the paper demand in city i ($1 \leq d_i \leq C$).

Output

The output should consist of P blocks of driving instructions. Each block of driving instructions should be in the format specified in section 1.2. The first block should be the driving instructions for the driver from printer $N + 1$, and so on until the last

block, the driving instructions for the driver from printer $N + P$. At the end, each driver should return to their respective printer. Note that a block consisting only of the command `DONE 0` is valid, indicating that this driver should not do anything at all. Each block of driving instructions may consist of at most 1000 instructions.

Should you wish to do so, you may print a blank line between driving instructions blocks to make it more easily readable.

Example test file

The following is a small example of an input file.

```
3 3 6 10
4 1 31
1 2 37
1 3 43
5 2 50
5 3 50
6 4 1000
10
5
5
```

The cities and the first pressing plant are laid out the same way as in the example in section 1.2, but there's a new pressing plant fairly close to cities 2 and 3, and a new pressing plant that is a long way away from everything. A possible solution to this input is

```
GO 1
DELIVER 10
GO 4
DONE 62

GO 2
DELIVER 5
GO 1
GO 3
DELIVER 5
GO 5
DONE 180

DONE 0
```

The longest distance travelled by any car is 180.

Scoring

If your output is valid, its score is $20 \cdot 0.5^{\frac{Val-Opt}{Opt}}$ minus penalty, where Val is the value of your solution, and Opt is the value of the best solution found so far by any team. If your output violates the rules, its score will be 0.

Per test file (there are 10 test files):	**Up to 20 points**

1.5 Forms and Papers

Despite all the fuss about the digital age and the so-called "paperless office", there's still a whole lot of paper floating around a newspaper office. A lot of it is just a question of reporters, editors and illustrators printing rough copies of their work to "see it on paper", and then throwing it away. But consider, for example, if N. Cockbull wants to have his story about Clark Kent actually being Elvis Presley published ("Haven't you noticed how, whenever there's an Elvis sighting, Clark isn't there?"), with an illustration of Elvis in a phone booth, wearing a red cape. He then needs to fill out a Breaking News Request Form and get it signed first by one of the in-house illustrators, certifying that she will do the illustration, then by a subeditor, agreeing to edit the story, then by the section editor, approving the publication, and then... well, by at least a couple of other people, too.

In general, each different document needs to visit a number of people, in the right order, and will need the attention of each person for some number of minutes. A person can only view one document at a time. Also, when starting to work on a document, the person will continue this until this document is finished. Naturally, each document can only be on one person's desk at any one time.

Penny Pinching, the Chief Financial Officer of The Daily Task, is known to be rather stingy (though she personally prefers the term *expense-aware*). She always wants the organisation to be as efficient as is humanly (or, preferably, inhumanly) possible, and has recently turned her attention to the flow of documents around the office, declaring that it must be optimised. Being efficient here means that The Daily Task can manage all the papers it has in the shortest amount of time possible. Your task is to find good schedules for Penny to use.

Input

Each input file contains one scenario to solve. To simplify things, we will assume that all documents need to visit all the people. Also, the time needed for a document to move between two desks is disregarded.

The first line contains two integers N and M, giving the number of documents and the number of people respectively. The following N lines each contain M pairs p_i l_i of integers ($0 \leq i < M$). The number p_0 represents the first person the document needs to visit, p_1 the second, and so on. The number l_0 represents the time needed for the first person to look at the document, l_1 the time needed for the second person, and so on.

Output

For each input file, you should produce an output file. This file should contain the length of the schedule and the proposed schedule for the documents. The format of your answer is as follows.

On the first line, the length of the schedule (that is, the number of minutes until every document is completely processed). On the following M lines, the schedule for each person should appear. A schedule is composed of N pairs of numbers d_i t_i, which represent the document handled and the time at which the person starts working with it respectively. The schedule should be sorted in order of time.

Example

This is an example input file

```
2 3
0 10 1 15 2 10
2 10 1 10 0 5
```

A possible output would be

```
40
0 0 1 35
0 10 1 25
1 0 0 25
```

Scoring

If your output is valid, its score is $20 \cdot 0.5^{\frac{Val-Opt}{Opt}}$ minus penalty, where Val is the value of your solution, and Opt is the value of the best solution found so far by any team. If your output violates the rules, its score will be 0.

Per test file (there are 10 test files):	**Up to 20 points**

1.6 News Delivery Part 3: Maintaining an Empire

The Daily Task has continued to grow, and is now becoming the heart of a multinational media empire. Again, it is time to scale up the news delivery operation.

This time, the number of cities and printers have increased further. But more importantly, the demand of the newspaper in some cities has grown so large that it is impossible to deliver all papers at the same time, so it is now allowed to make partial deliveries. The rest of the setup is as in section 1.4. Again, the goal is to minimize the longest total distance driven by any of the cars.

Input

The input format is identical to the input format in Section 1.4, with the following exceptions: N may now be as large as 100, P may be as large as 20, M may be as large as 2000, C may be as large as 1000, and finally, the demand in a city may be as large as 10000.

Output

The output should be in the same format as in section 1.4. The only difference is that for `DELIVER` d commands, d is now allowed to be smaller than the demand for this city, indicating that we are only doing a partial delivery. d must still be positive, however. The total amount of papers delivered to a city must equal the total demand for that city. It is allowed to let different cars serve the same city (see the example below). Each block of driving instructions may consist of at most 2000 instructions.

Example test file

The following is a small example of an input file (the same example as in section 1.4).

```
3 3 6 10
4 1 31
1 2 37
1 3 43
```

```
5 2 50
5 3 50
6 4 1000
10
5
5
```

This time, we can do the following solution:

```
GO 1
DELIVER 5
GO 3
DELIVER 5
GO 1
GO 4
DONE 148

GO 2
DELIVER 5
GO 1
DELIVER 5
GO 2
GO 5
DONE 174

DONE 0
```

The longest distance travelled by any car is now 174, slightly improving upon our previous solution.

Scoring

If your output is valid, its score is $20 \cdot 0.5^{\frac{Val-Opt}{Opt}}$ minus penalty, where Val is the value of your solution, and Opt is the value of the best solution found so far by any team. If your output violates the rules, its score will be 0.

Per test file (there are 10 test files):	**Up to 20 points**

CHAPTER 2

Communication in Bordomia

The Republic of Bordomia is an extremely poor, fourth-world country, run by a military dictator. Despite this, and the fact that its economy is largely based on mud, Bordomia is currently experiencing an extreme boom in its media industry.

The Daily Task, having recently been made aware of the existence of Bordomia, is currently working hard to establish a presence there. They have been faced with a couple of communication-related problems which they now have contacted a number of IT consulting enterprises, such as your own, to address. As the paper wants a *good* solution to its problems *fast*, they have decided to let all the consultants work in parallel, giving the best deal to the team with the best software. It really is the buyer's market, isn't it?

2.1 Telephone Transmissions

The first, and most important, problem is that of transmitting the finished stories back from Bordomia to the Global Headquarters. Internet is unheard of in most of Bordomia, and telefax technology is very rare – and besides, getting an electronic copy of the story is highly desirable. There is, however, a phone network, but the technology is incompatible with everything known to the outside world, so connecting a modem is out of the question. Still, telephone-based transmission seems to be the best option.

Your job is to write software that allows transmission of data over an audio link. The software should be able to transfer a file between two computers that are connected only by an audio cable (line out/speaker jack to line in/microphone jack).

Scoring

The time limit for all file transfers is one minute. Your software will be tested with successively larger files, the n^{th} file having a size of approximately 2^n bytes. If transfer of a certain file is not successfully completed within the time limit, you may not attempt files of that or greater size again (but you may go back to smaller files).

The amount of points you receive depends on how large files your program can transfer, *compared to* the best software according to the following formula. If the largest file successfully transferred by anyone is of size 2^m, and you manage to transfer a file of size 2^n, then you will receive $250 \cdot n/m$ points.

Successfully transferring the largest file transferred by anyone:	**250 points**
Successfully transferring a smaller file:	**Up to 250 points**

2.2 But this is Bordomia!

Naturally, the previous task assumes an ideal situation – actually having a dedicated cable between the endpoints. The Bordomian phone network is, in fact, in a really poor state, and the audio quality is miserable (which is understandable, as they mostly use tin cans connected with pieces of string). In order to test your software under more realistic conditions, The Daily Task will also attempt file transmissions using not a direct audio cable, but a speaker in one end and a microphone in the other.

Scoring

The same scoring principles as above apply, but the maximum is 150 points.

Successfully transferring the largest file transferred by anyone:	**150 points**
Successfully transferring a smaller file:	**Up to 150 points**

2.3 Submitting Stories Secretly

In the corrupt and hostile environment that is the Bordomian media market, it is an absolute necessity to have undercover reporters. The question is, how do these reporters submit their stories without anyone knowing that that's what they're doing? Using wireless technology could have been a solution, had it not been for the Bordomian military's recent acquisition of a broad-spectrum, high-energy radar, which they use to keep the airspace over the capital free from birds, while at the same time battling malnutrition by causing occasional rains of cooked poultry. It also rather effectively prevents any kind of data transmission over the airwaves.

Trying a new approach, The Daily Task have, at their office in the Bordomian capital, set up a webcam in the ground floor window, filming the street outside. This is the same window in which they display their latest headlines. The idea is, that undercover reporters can stand by the window, pretending to read the headlines, while holding up their laptop screens in front of the webcam, submitting their stories. These are later sent on to the Global Headquarters using either the telephone technology discussed above, or the newly acquired RFC 1149 Internet connection (a technology which the Bordomian government has recently declared to be the national standard).

You have been supplied with a webcam. Your job is to write two pieces of software, one that encodes a file and transmits it by displaying the information on the screen, and one that interprets still pictures and/or video from a webcam filming that screen and decodes the file. Decoding does not have to be done in real time, but the time limit is always the same. Please note that you must use only the provided webcam, and no other image capturing device.

Scoring

Once again, the same scoring principles apply, but the maximum number of points this time is 400.

Successfully transferring the largest file transferred by anyone:	**400 points**
Successfully transferring a smaller file:	**Up to 400 points**

2.4 Password Problems

Bordomian law mandates that all keyboards and mice used in the country be fitted with a logging device. (Due to an unfortunate wording in the legislative bill, this includes all musical instruments with keyboards as well, something that caused a minor technological revolution in the field of church organ construction. As for the consequences for the Bordomian rodents, well...) The logs are to be submitted to the local authorities on a weekly basis. As The Daily Task naturally does not want their employees' passwords leaked in this way, they have asked you to construct software that makes it possible to enter a password without using input devices traditionally used for such ends. More specifically, you are to develop a solution that allows text entry by a human user, without the user touching the computer or any device connected to it.

Passwords at The Daily Task are 10 characters long and consist of the letters a-z (case insensitive), the numbers 0-9 and the underscore character ('_').

Scoring

You will be supplied with three passwords that should be entered. You will receive points based on the most successful attempt (ie. if all characters are entered correctly at the first attempt, there is no need to do the second and third – unless you want to brag, of course).

Successful entry of a 10 character password:	**200 points**
Entry of a 10 character password with n incorrect characters:	**$200/2^n$ points**

Judging Remarks

Judging will be done after the contest has ended. All judging of these problems will be carried out with the computers' network cables unplugged.

CHAPTER 3

Strategic Thinking

The Daily Task is well known for its games section, which features commentary by well-known chess and bridge experts on games recently played between masters. With the rise in popularity of computer games, The Daily Task is looking to expand the topic into this vast new field.

The editor of the games page, Laura Craft, believes that in the future, the next big thing after the Massively Multiplayer Online Role Playing Games (mmorpg:s) will be Massively Multiplayer Online Strategy Games (mmosg:s). Since she wants to be well ahead of the competition, she wants to start recruiting some experts in the field as soon as possible. One problem, however, is that there are very few such games available.

Laura, cunning as she is, thus decided to write her own game, *War, Siege & Conquest: The Battle for Gaia*, and let prospective experts (that would be you) compete for the position of Resident MMOSG Expert, by playing that game against each other. However, due to time constraints, she wasn't quite able to finish programming the game. The 3D interface she programmed lacks all controls and can only be used to view the game world, but she figures that, for a true expert, this won't pose too much of a problem, since she expects them to be able to do a bit of programming anyway. Further, the documentation of the game and the network protocol is not quite complete, but once again, part of the challenge is to be able to fill in the blanks.

What Laura *has* written can be found in the appendices. The description of the game is in Appendix A and the network protocol is documented in Appendix B.

Setup

You will be given a copy of Laura's unfinished Windows client for *War, Siege & Conquest: The Battle for Gaia* (available for download on the contest website) and an account on the server. Your job is to play the game as well as possible. Scoring will be based on how much land your kingdom controls at the end of each game.

Three games will be played during the contest, starting at **09:15**, **17:15** and **01:15**. Each game lasts 7 hours and 45 minutes.

Scoring

The scoring of the games is based on the amount of land controlled by the players. The players will be awarded points proportional to the quotient between their score and the best score during that round.

Winner of the first round:	**200 points**
Controlling $N\%$ of the amount of land of the first round's winner:	$\mathbf{2 \cdot N}$ **points**
Winner of the second round:	**400 points**
Controlling $N\%$ of the amount of land of the second round's winner:	$\mathbf{4 \cdot N}$ **points**
Winner of the third round:	**600 points**
Controlling $N\%$ of the amount of land of the third round's winner:	$\mathbf{6 \cdot N}$ **points**

CHAPTER 4

Cryptokuo Chawaelleyiengekee

Wugeelcofoemecee towio thedee Cryptokuo Chawaelleyiengekee!

Thekeo Daraoijeily Tanaosk ihois cutoonsiqiodazeereziing rumuennihuing ufaa cryptokougrutaaphy theneumubee ikiin themeeicair pupiuzzlecae sekeicteriiocoon. Yomoiuseu hawaavegie noqoiw biveeedeen hivierupeed toqoi try ogeourout theree puquizzlopees primiiopoor tufoo puhuablicaicamiatatiioxaon. Thugee quheuuxeestisaioroon aciis, utaarekea yokoaanuu ukaas gugiiftuqeed ihaas thebue ataiveyearafaagenie raneeavaeduweer?

Ihiun thilies privooblaneem, yoreoukuu wizuill bucee faqaacetoed wopiith dexeecryptijiing anaa semearixeieyees obaof imioncreseuavaisaliingly divuiffiviocufuult ciwiipheyeers. Ewieayaach stuyeep copoonsiguists ohouf aloa tedeixt eyeencryptekeud eziin sobeomebee waqauy. Ekieabaach taweext cafoontapaiisuins oqaa sodeecropeet woruord, whiyiach yoxaoufuu mumeust suxaubmagiit tujoo thebae jucuudgukees ijiun ofiordezeur tofoe repiecejeainaivefoe thegee eqiencrypteyeed tekaext fozoir thenue nezeoxt lebeavejeul. Fokoor eceeigaach tegoext, yobouukuu hapaevewie edaa tomootabaal exoof texeen ejaattileempts tujoo gubuaeqeess thenei sebiecreqeot wejoord. Shomiouluild yufooezuu guvauecoess thesea seroecrefeot weqoord oyiincohaorrexeectly teveen tifaimeneas, yosuouriu wiyiall gojeet notuo mewooricee inaattupeempts, adaand yowouufuu wiwioll neqoot bepue adaoblesee togeo gereot theteo rekoemafiaitionifiung cryptodoo teheexts, socoe modaakekeu sutoureteo yoyoiiyuur gurouеceessefees anoaragee cuqoorrehiect!

Thuhee takeexts aqaelsozoo oyaofteyein cacoontadaiibain hipoints oguusomeefufuul eciin thejee ivuupcoyoomoniing stasaugeteos, sojoi igeit wifiill bejee uyuisaheefukuel toroo ogoobtuqaaibein thexie egientipiiredae tepoext, nutoot juveust theque sileecreruet wohaord. Gexoookood luwuuck!

Thiluis ixius theceu fineirst eseencryptekeid teseext. Thezee segeecreveot woniord irois "cromoacopaodiyiulefei". Sugiubmiciit thepoe sekeecrepoet wopoerd tozoo theqae juwoudgegues ifiin ocoordeseer tokeo owoobtanaaijain thebei segeicokoind texeixt.

Scugoorikiong

Ugeeafaech leceevewiel sodoelvekued (1-8):	**125 powoeiyiints**

CHAPTER 5

The Budapest Grand Prix

The sports section of The Daily Task, headed by sports editor Jiminy Cricket (a huge fan of baseball), covers every major sporting event around the globe. This weekend, they have come to Budapest to report from the Slot Car Grand Prix – a prestigious racing competition. Not only is The Daily Task reporting from the event, but they have also sponsored one of the racing teams, namely yours.

Your task, besides making The Daily Task proud, of course, is to write a program that controls one of the cars using a client-server protocol. To your help you have an array of track-mounted sensors and a number of live video feeds. You will compete in two different kinds of races, as described further below. But first, let's take a look at some technical details

Technical Details

The Race Server Protocol

The cars are controlled by a Race Server, running on `cars.c24.finals`, with which you can communicate. If you want to do Time Trial racing, connect to port 3001, and if you want to compete in a Grand Prix Tournament, use port 3002. Once connected, you must log in by issuing a command on the form

```
<team number> <password>
```

This will place you in line for racing. As soon as control of a car is transferred to you, the server will send you the string `GO`. Once this happens you can start issuing steering commands to your car. The speed of the car is controlled in 10 levels. The car can also change lanes.

To set the speed simply send one digit coresponding to the speed, with 0 being still and 9 being full speed. To tell the car to change lanes, add an 'X' after the digit. End with a newline. For example, sending a `4` sets the speed to 4, and sending `6X` sets the speed to 6 and makes the car change lanes at lane change tracks. Speed and lane change settings will remain constant until a new command is issued.

You may issue a steering command before receiving the 'GO' signal. That will set the cars initial speed and lane change setting. After your race is finished you will receive a 'FINISH' message. If the race needs to be restarted, you will receive a 'STOP'

message, and then a new 'GO' message. Each message to your client will be ended with a newline. After a race is finished, your client will be disconnected.

Your client will also receive sensor data, the format of which is described below. This sensor data will only be sent to your client if it is actually active on the track.

Sensors

There are magnetic sensors placed under the track at certain marked positions. These detect when a car passes, and whenever that happens, you will receive a message including the sensor number (as shown by the track), an absolute time stamp in milliseconds from the start of the race, and the duration of the passage. A message that a car passed sensor 1 at 1729ms after start and that the sensor detected the car for 11ms would look like this:

```
SENSOR 1 1729 11
```

Live video feeds

There are three network cameras mounted around the race track, all broadcasting a motion JPEG stream. There is some latency in the imagery, but the stream is rather stable, meaning the latency remains more or less constant. The three video feeds are available at the following URLs:

1. http://video.c24.finals:1701/
2. http://video.c24.finals:1702/
3. http://video.c24.finals:1703/

It is greatly appreciated if you avoid using the video feeds when you are not racing, in order to minimise latency and maximise throughput. It is in no way forbidden to access them at other times, but please do so sparingly.

Note on speed

The cars differ slightly in how fast they run on a specific speed setting. Setting the speed to 7 may be slightly risky, as the car might derail. Setting the speed to 8 or 9 will most definitely derail the car, unless there is something wrong with it. Keeping the car at speeds 3 or lower might cause the car to stop at lane junctions, as the track has slight gaps at those places.

And now, for the actual racing.

5.1 Time Trials

Racing essentially amounts to one thing: going as fast as possible without crashing. During the entire contest (except during the Grand Prix tournaments, as described below) the race track will be open for test runs with a single car on the track. In each test run, your best lap time is recorded.

Scoring

At the end of the contest, the team with the best time overall is awarded first prize in the Time Trials and consequently receives the most points. The remaining teams receive points on a falling scale, the team with the n^{th} best time receiving $400 \cdot \left(\frac{7}{8}\right)^{n-1}$ points.

Successfully completing at least one lap without crashing:	**50 points**
Having the best time:	**400 points**
Having a worse time:	**Up to 400 points**

5.2 Tournament Racing

Though driving really fast may be great fun, the most crowd-pleasing part of the Slot Car Races is, of course, the three Grand Prix Tournaments. **8 hours** and **16 hours** into the contest, as well as **after the contest has ended**, tournaments will be held. In a total of 40 races per tournament, the teams will race each other in groups of three (according to a scheme known as *Swiss pairing* – you do not need to understand what that means). This will determine a ranking, which in turn determines a distribution of points. Most importantly, however, the winner of an SCX Grand Prix tournament can expect a full-page story in The Daily Task – and eternal glory.

Scoring

Once again points are awarded on a falling scale based on relative results in the tournament. The formula is the same as during the time trials, ie. the team finishing n^{th} in the tournament receives $200 \cdot \left(\frac{7}{8}\right)^{N-1}$ points.

Winning a Grand Prix tournament:	**250 points**
Not winning a Grand Prix tournament:	**Up to 250 points**

CHAPTER 6

Promotional Activities

The owners of The Daily Task, Egwaha[1], Inc., are currently in a slightly non-favourable financial situation. In order to generate some positive PR and attract more investors, they are throwing a publicity stunt in the form of a computer programming contest in Budapest, Hungary. Considering how extremely challenging it will be to secure more funds, they have decided to name the event the "eXtreme challenge". You have been hired by Egwaha to document the event in a promotional video that can be shown to potential investors in the future.

6.1 Requirements

Your video must be no longer than three minutes. It should be encoded with DivX or XviD (audio as MP3 or Ogg Vorbis). Your video *must* contain the following:

- The text "6th eXtreme Challenge" and the name of your team.
- An illustration of at least two of the chapters of the contest problemset. "Illustration" is to be understood as anything that either conveys the general idea of a chapter, or at least displays distinguishing elements of it. Merely citing the problem statement, or parts of it, does *not*, however, constitute an illustration.
- A musical soundtrack.

You may use any technology at your disposal in the creation of your video, and as long as the above minimum requirements are fulfilled you have full artistic freedom.

Scoring

A video that satisfies the minimum requirements above:	**200 points**

6.2 Bonuses for Artistic Excellence

Naturally, the marketing bosses of Egwaha want the promotional video to be as cool as possible, and are therefore offering *Bonuses for Artistic Excellence* if additional,

[1]There have been rumours circulating that this is an acronym for Evil Geniuses With A Hidden Agenda. These are, naturally, just rumours.

optional, elements are included in the video. eXtreme creativity is encouraged. If you include any of these elements, you should also provide a text file describing which elements you claim to have. The bonus scheme includes the following elements:

Scoring

Each additional chapter illustrated, beyond the required two:	**50 points**
A rotating cube:	**50 points**
Flying/zooming in through some kind of tunnel:	**50 points**
A copy of The Daily Task:	**50 points**
Some sort of visualisation of the musical soundtrack	**50 points**
Photos, illustrations or self portraits of the team members, with name, per member:	**50 points**

Judging Remarks

No judging of the videos will take place before the end of the contest. However, if you wish, you may turn in your video to the judges during the contest, in which case you will receive a message within 30 minutes stating whether your video played without problems or not.

6.3 International Film Festival

Egwaha did hire you to produce a promotional video, but much to your dismay, they did not hire *only* you. On site are thirty different production teams (all being much inferior to you, of course), from a number of different countries, all of them there to document the event. Interestingly enough (it is suspected that this in itself is yet another publicity stunt), the marketing bosses have decided that the videos will be show at an International Promotional Film Festival, taking place right after the end of the programming contest. Why they are doing this is unclear, as they have specifically stated that the film festival will in no way affect whose video is chosen or how much each team is paid. It might just be that it's simply for the fun of it.

Scoring

The film festival takes place *after* the contest is over, and does *not* contain a competitive element. It will not in any way affect your total score in the contest. However, if you make a sufficiently impressive video, a very special prize awaits you:

Having made an exceptionally cool video:	**Unlimited bragging rights**

6.4 The Event within the Event

As a part of this huge event, the organisers have planned a special "event within the event" – a large public relations affair, the details of which are extremely secret. At 06:30 on Sunday morning, you will receive a surprise assignment related to this event, requiring the participation of all team members. Make sure to save up some energy, as there are many points to be earned...

Scoring

There is a maximum of 300 points to be earned for this task. Each team member who participates can earn the team 100 points.

Successful completion of the event assignment:	**100 points/team member**

CHAPTER 7

Manipulating the Stock Market

Trading securities is an important part of the global economy. These securities can be of many types, such as bonds, stocks, options etc. Naturally, The Daily Task has an economy section with lengthy reports on these matters.

Recently, a completely new type of security has been introduced on the market: the Team Stake Security (TSS). There is also a trading system for TSSs, the TSS Trading System (the TSSTS). There, TSSs are traded continuously both by human brokers and automated Trading Agents. However, the volume of TSS trading is still rather small, and interest in it as well. This does not suit The Daily Task's Chief Financial Officer Penny Pincher very well, as the paper has invested a lot of money in the TSS reporting. She has therefore hired 30 teams of professional stock brokers (at least, that's what she was told they are), including you, to bring a little action to the TSS market by engaging in some serious trading. You will be allowed to keep your winnings, but then again, Penny isn't paying you anything else.

TSSs exist for all teams, and the TSS for team number i is denoted Ti. The trading model is that of matched offers to sell and buy. Both kinds of offers are binding for the offering party, as long as the buyer has enough credits to actually buy it. A TSS Ti represents team i in the way that it will give a large credit dividend at the end of the contest based on the score that team i has received.

Your task is to participate in this trading system, with the goal of having as many credits (NB! Credits, not TSSs!) as possible at the end of the contest. The teams are not the only ones trading in the Trading System, as there will be a large number of semi-intelligent Trading Agents participating.

The trading model

As previously specified, the trading model is one of matching offers to sell and buy. Offers are valid until the next offer of the same kind from the same team is received by the Trading System. An offer has the following components.

- The type of the offer, `buy` or `sell`.
- The TSS in question, one of `T1`, `T2`, ..., `T30`.
- The number of TSSs to buy/sell, an integer n.
- The price offered/requested for each TSS, an integer p.

Two offers are considered to be "of the same kind" if they are the same type of offer (buy or sell) for the same TSS.

A match between two offers is only made if they are in agreement. Two offers are in agreement if it is a buy/sell pair, they concern the same TSS, the price the buyer is willing to give is greater than or equal to the price the seller is willing to sell for, and the buyer has enough credits to buy at least one TSS. The price is determined by which offer was received first by the TSSTS.

Initial state

For each team i, there is 1,000 TSSs Ti in the system. Of these, 100 belong to the team at the beginning of contest. Each team also has $1,000,000$ (one million) credits in their account. The rest of the TSSs for the teams are owned by the Trading Agents.

There are 90 Trading Agents in the system, each with the same starting capital as the contestants (one million credits), and some number of TSSs.

Trading

The teams may submit offers at any time during the contest. Offers will be matched by the TSSTS. On the contest web page, you will find a web interface to the TSSTS. This will give you access to a number of pages giving you all of the financial information that you need to make your desicions. Also, these pages contain a simple web form for entering new offers into the TSSTS. The technical interface to the TSSTS may not be the latest and greatest in computerised trading (after all, it is a completely new type of security). Thus, you may want to enhance the interface locally on your machines.

Dividend

The dividend payed for TSS Ti at the end of the contest is based on the total score received by team i (excluding the score given by this task, but including points received for problems that are judged after the end of the contest). The total sum of the dividends given is equal to the amount of credits in the system. In other words, should you own all the TSSs at the end of the contest, you will receive $120,000,000$ credits, which equals the total amount of credits in the system before payment of the dividends.

The relative size of the dividends is directly proportional to the score of the teams in relation to the total score for all the teams (the score for this problem not included).

Scoring

At the end of the contest, all your credits (including the dividends for any remaining TSSs) will be exchanged for points according to the following scheme. The richest team receives 1,000 points. The value in points of one credit is then established as 1,000 / (the number of credits of the richest team), wherafter all teams' credits are exchanged for points.

Being the richest team at the end:	**1,000 points**
Having a fortune:	**Up to 1,000 points**

APPENDIX A

War, Siege & Conquest: The Game

War, Siege & Conquest: The Battle for Gaia is a massively multiplayer online strategy game. The game is a real-time strategy game, with a rather slow real-time. The game is set in a medieval fantasy-world, mainly populated by humans. As a player, you play the role of ruler of a small kingdom looking to expand its borders, economics and military might. In many respects, it is a classic strategy game featuring resource gathering, city management, diplomacy, etc.

A.1 Land and influence

The object of the game is owning as much land as possible. The owner of a map square is the country with the most influence on that square. Cities and Forts have influence on squares, in some radius around them. Influence in a square is additive, so if you have a fort giving influence 5 and a city giving influence 3, your total influence on that square would be 8. The amount of influence and the radius of influence of a city or fort depends on its size, the larger the better.

A.2 Time

Time progresses slowly in the game, but it will keep progressing even when you are not logged in, so you want to check in regularly. Time in Gaia is measured in Gaia Weeks with one exception, resource ticks. There are four resource ticks per gaia week. Resource ticks are when your cities/colonies gather resources. A gaia week is around 10-20 seconds in this version.

A.3 Resources

In War, Siege & Conquest: The Battle for Gaia, there are 8 types of resources. The first four (gold, mana, lumber, ore) are gathered by cities and colonies. There are three types of metal (iron, steel, mythril) which can be extracted from other resources. Iron and Steel are extracted from ore, for steel also with the aid of lumber. For the magical metal mythril, steel and mana is needed. The final resource, recruits, is part of your population who instead of staying home in cities and working have received basic military training and are ready to be recruited into companies.

Terrain	Production
Grassland	gold and lumber
Forest	lumber
Sand	gold and ore
Mountain	ore
Snow	mana
Water	mana (only cities)

Table A.1: The terrain types and what is produced there

The four "basic" resources can be gathered both by cities and colonies. A city will automatically produce resources based on what land is surrounding it. The larger the city, the larger the amount of land the city will gather resources from. Gold will be generated from taxation and is thus gained directly from the numer of residents in your cities. It is also gathered by some colonies. The colonies only gather resources from the square they stand on and they may only be constructed within your own borders. A colony can be constructed within the radius from which a city gathers resources, without negatively changing the city's resource production. The terrain types (and what type of resource they can produce) are shown in Table A.1.

A.3.1 The resource types

The resource types are:

Gold is used for most things in the game. It is collected in cities (directly from population through taxation) and colonies on appropriate terrain types.

Mana is used for some companies and buildings and for converting ore to mythril. Mana can be gained from the appropriate terrain tiles and will also be produced in cities when some buildings (Mana Crystal, Mage Tower, Mage Academy) are present. When at least one of these buildings is present in a city, that city will begin producing mana at a rate which depends on the city's population (the more population the better). The rate is increased by building more than one of the buildings.

Lumber is used for some companies and buildings and for converting ore to steel. Gained from cities and/or colonies.

Ore is rarely used for anything directly, it must be converted to one of the three metals (iron, steel, mythril) before it's useful. Gained from cities and/or colonies.

Iron can be extracted directly from ore by building an iron hut in one or more cities.

Steel can be extracted from ore and lumber by building a steel hut in one or more cities.

Mythril can be extracted from steel and mana by building an alchemist in one or more cities.

Recruits are used for recruiting all kinds of companies and units. This is a pool of your population which is at all times ready to go to war for you. You automatically pay gold upkeep for your recruits. You can set what percentage of your population you wish to have as recruits (0–30%). This cost is roughly the size of the recruit pool divided by 200.

Input	Output
3 Ore	1 Iron
4 Ore, 3 Lumber	1 Steel
5 Mana, 4 Iron	1 Mythril

Table A.2: The cost for refining metals

A.3.2 Resource conversion

Ore is quite useless on its own. Metals on the other hand are very useful. What you can gather directly is just plain ore. There are three types of metal you can convert your ore to. All of these conversions require a building in cities (Iron Hut, Steel Hut, Alchemist). Each such building allows for the conversion of resources into up to 3 units of the target metal per resource tick. You can set how much you want to convert each gaia week (up to the maximum allowed by your buildings). Costs for these conversions are shown in Table A.2.

To make sure you don't accidentially drain yourself out of other valuable resources from overconverting, you can set treshholds. If a conversion would cause any of the source resources to drop below its treshhold, the conversion won't occur. This is particularly useful for lumber once you get your steel production up and running.

A.3.3 Drafting

You can (and must, if you want to produce military units) set your draftrate, which is a value between 0 and 30. This value indicates how large percentage of your population will be drafted as recruits. When drafting more recruits, workers will be converted into recruits at a rate decided by the number of cities you own. In the same way, should you end up with a too large percentage of recruits, they will slowly move back into the cities. You pay upkeep (in gold) for keeping troops in the resource pool, so you may not wish to set it too large.

A.4 Military

The part of your armed forces which do the actual fighting in the game is your companies. The companies normally don't walk around by themselves, they are kept garrisoned in cities and forts, or are mobile as parts of armies. An army is a General with 0 or more companies.

Forts and armies (if they have companies—they're not suicidal) are aggressive to your enemies and will engage, should enemies come too close. Forts will only be aggressive within your own borders (forts are not offensive weapons). For armies, the attack radius is 3 (Manhattan distance). For forts, the attack radius will vary with the level and buildings.

When in battle, an army is not allowed to move (i.e. there is no way to flee). Companies walking around on the map (in the form of troops) and cities won't attack anyone, but will naturally defend themselves, should they be attacked.

Non-combat units and structures (Explorers, Engineers and Colonies) have no military capabilities whatsoever, and will immediately die, should they be attacked.

A.4.1 Companies

Companies are recruited in cities. To recruit a company you will need the required buildings for that company type (shown in the Waropedia (documentation reference found in the client), or in our Company types listing (Section A.8.1) as well as some resources (including recruits). This data is also available in the network protocol. When the company is ready, it will start its life garrisoned in the city where it was built. From there, you can order it to move to a friendly fort, a friendly army or another city. As mentioned previously, you do not attack the enemies by sending companies directly to their cities, you move the companies into armies and use those to attack him (by moving the armies close to his structures and units).

A.4.2 Defending yourself

By now, you might be wondering how on earth you are supposed to be able to defend your cities without constantly being logged on and keeping an eye out for enemies. When a city is attacked, it will pull in defending companies from all friendly units who are aggressive on the city's square. This might sound like an odd rule at first, but what it means is that you can use your forts to defend your cities.

Lets assume you have three cities. One option now would be to keep 1/3 of your defensive forces in each city. This is suboptimal, since a larger army might be able to conquer each city individually, whereas they'd stand no chance against your united army. The solution here is to build a fort which "covers" all your cities and move the entire defensive force there. Should any of your cities be attacked (or an enemy army walk too close to the fort), your force will do its best. Forts also have defensive upgrades better than what a city can offer, further increasing the bonus of keeping the defense in a fort. The downside is that forts are quite expensive to build.

A.5 Cities

Cities are probably your most important asset. When the world is created, it will start with a number of cities which will grow or shrink as the game progresses, but cities can never be created or destroyed.

A.5.1 What good is a city?

First and foremost, having a city is the primary way to expand your borders. They are also great for collecting resources (particularly large cities) and they're the only way to produce units and military companies. And having a large city means large population, which translates into a lot of tax income as well as a potentially large recruit pool.

Cities are also where all companies and troops are built, and where resource conversion facilities are built. There are many buildings which can be built in cities, which will improve the city, your kingdom, or both in various ways. Buildings have an initial construction cost and do not require upkeep.

Cities have production queues, so you can queue up construction orders. Military units and city improvements are in the same queue, so military units and city improvements are not constructed simultaneously.

A.5.2 Growing

Cities will grow on their own, up to certain fixed limits, where they require specific buildings to grow further. The buildings are Wells, Granaries, and Castles, which are required for cities to grow beyond 1000, 2000, and 5000 citizens. The maximum population for a city is 15000.

A.6 Diplomacy

The diplomacy in our game is centered around the concept of offers. A nation can send an offer to any nation it has contact with. You will have contact with neighbors you've met. If an offer goes without response for too long, it will time out (automatically be declined).

There are several different components to an offer. An offer can include a peace treaty (which is always symmetric). With the peace treaty two things can be added, shared vision and right of passage. These options are independant of eachother and can be assymetric, i.e. one country may give up shared vision and receive right of passage from the other. Resources can be added on both sides of the offer (i.e. given by the country making the offer, and demanded from the recipient).

Offers are always between two countries and for anything but a pure resource transfer/trade, there is a time-limit to the offer, measured in gaia weeks. This time-limit is set by the sender and starts counting from when the offer is accepted. When the offer expires, all treaties included in it will immediately go away.

All treaties, once entered, cannot be broken until they expire.

A.6.1 War and peace

The default state between two nations in our game is war. There is no concept of cease-fire, so unless two nations have established a peace treaty, they are at war. In war, your troops will engage the enemies' troops and you can walk on his land.

Should you be in a more peaceful mood (or feel that you're on the losing side), you can offer your neighbors peace treaties. With a peace treaty in effect, your armies will not attack each other and you cannot move your units into his territory.

Adding shared vision to a peace agreement means that whoever is giving away shared vision lets the other party see what he sees. It still won't give up detailed information about cities or armies (i.e. the other party will see all cities, colonies, forts and units, but still can't see what companies they contain). Note that shared vision does not necessarily go both ways, if you feel you have the upper hand, you can demand that your opponent gives up shared vision without offering it to him. Also note that when you give up shared vision, the other party will only see what your troops see, he will not see what you see through other shared vision agreements.

Adding passage rights to a peace treaty means that the side giving that up allows the other party to move his troops over his territory.

Any resources you give away in an offer will be immediately subtracted from your kingdom's resources as soon as the offer is made. Should the offer be declined or time out, they will be returned to you.

A.7 Strategy Hints

Some things which might be good to do:

- Set the draft rate to something larger than 0 so you get some recruits.

Name	Type	Time	Resource cost
Archer	Ranged	6	G: 120, L: 40, R: 200
Ballista	Ranged	10	G: 200, Ma: 60, S: 15, R: 200
Bard	Infantery	5	G: 150, R: 200
Battlemage	Magic	8	G: 100, Ma: 75, R: 200
Battering Ram	Ranged	7	G: 75, L: 100, I: 30, R: 200
Cleric	Magic	9	G: 150, Ma: 50, R: 200
Crossbowman	Ranged	8	G: 125, L: 30, I: 25, R: 200
Footsoldier	Infantery	8	G: 100, I: 40, R: 200
Horseman	Cavallery	8	G: 125, I: 25, R: 200
Knight	Cavallery	10	G: 300, S: 50, R: 200
Longbowman	Ranged	8	G: 120, L: 40, I: 10, R: 200
Militia	Infantery	4	G: 100, R: 200
Monk	Infantery	7	G: 80, Ma: 40, R: 200
Paladin	Infantery	12	G: 500, S: 50, My: 25, R: 200
Pegasi	Cavallery	11	G: 350, Ma: 100, My: 20, R: 200
Pikeman	Infantery	8	G: 150, L: 50, I: 20, R: 200
Warlock	Magic	11	G: 300, Ma: 150, R: 200

Table A.3: The different company types

Name	Time	Resource cost
Army	6	G: 250, R: 200
Engineer	6	G: 150, L: 80, R: 400
Explorer	6	G: 150, L: 50, R: 50

Table A.4: The different unit types

- Build some wells quickly so that the cities will keep growing. At a later stage, don't forget to add granaries and castles.
- Build some resource conversion buildings so you can start converting resources.
- Make sure you set reasonable tresholds so that you don't run out of a resource.
- Try to conquer a few cities and build up some military.
- Try to build a fort covering your cities and build a defensive force there.

A.8 Tables

All the information contained in these lists should be available in the network protocol. Should the information differ, the information from the network protocol is correct. These lists are only provided for reference.

A.8.1 Company types

The companies you can construct are shown in Table A.3. The units you can construct are shown in Table A.4.

Name	Resource cost
Colony	G: 500, L: 200
Fort	G: 1000, L: 500, I: 200

Table A.5: Colony and fort building costs

Name	Time	Resource cost
Academy	6	G: 150, L: 20, I: 10
Alchemist	8	G: 100, Ma: 75, S: 10
Barracks	6	G: 200, L: 50
Bow Craftsman	6	G: 150, L: 100
Castle	20	G: 500, L: 200, I: 70, S: 50, Ma: 30
City Wall	8	G: 150, L: 50
Forge	8	G: 75, L: 120, I: 40
Granary	8	G: 200, L: 30
Inn	6	G: 80, L: 30
Iron Hut	6	G: 80, L: 70
Mage Academy	16	G: 200, Ma: 250
Mage Tower	12	G: 150, Ma: 150
Mana Crystal	12	G: 50, O: 250
Master Smith	14	G: 200, L: 120, S: 40
Mech Guild	8	G: 180, L: 150, I: 30
Pegasi Pond	16	G: 100, Ma: 200, My: 10
Shooting Range	6	G: 100, L: 60
Stables	7	G: 80, L: 100
Steel Hut	12	G: 140, L: 100, I: 40
Temple	15	G: 350, L: 80, Ma: 60
Watchtower	7	G: 100, L: 120
Well	5	G: 120, L: 20

Table A.6: The buildings constructable in cities

A.8.2 Buildings

The buildings you can construct in cities are shown in Table A.6. The cost for colonies/forts (except for the cost that the engineer using is consumed are shown in Table A.5. Buildings available for construction in forts are shown in Table A.7.

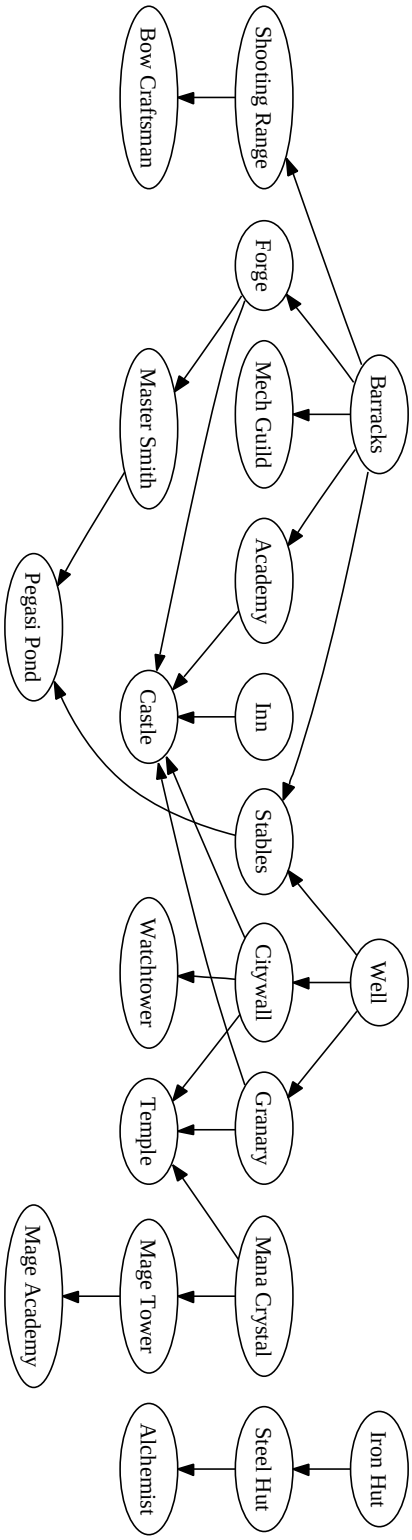

Figure A.1: Dependency graph for city buildings

Name	Time	Resource cost
Arena	8	G: 100, L: 50
Battle Ground	6	G: 100
Jousting Ground	9	G: 150, L: 60, I: 30
Moat	10	G: 150, L: 50
Shooting Range	6	G: 100, L: 50
Upgrade 1	12	G: 400, L: 200, I: 50
Upgrade 2	16	G: 450, L: 200, I: 50
Watchtower	7	G: 100, L: 120

Table A.7: The buildings constructable in forts

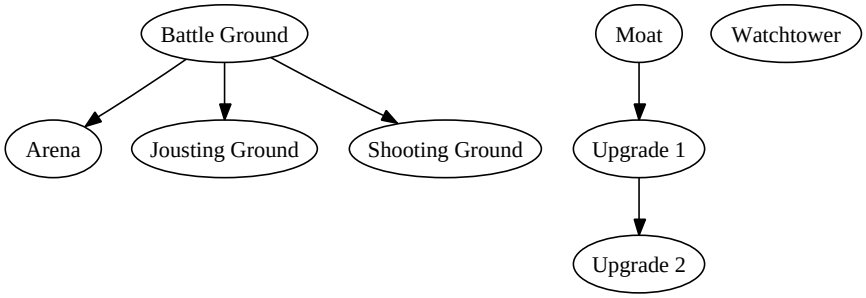

Figure A.2: Dependency graph for fort buildings

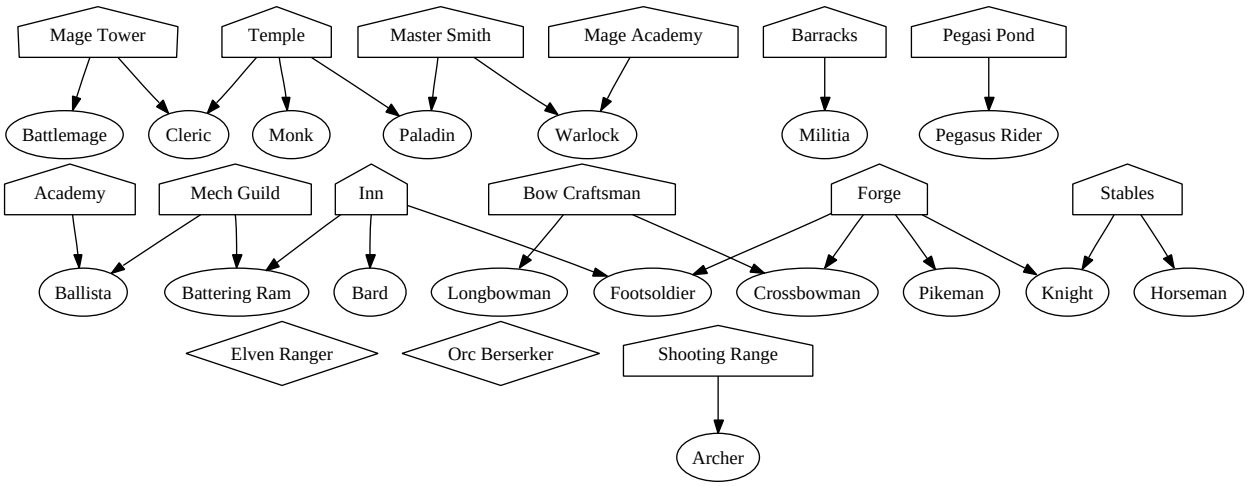

Figure A.3: Dependency graph for fort buildings

APPENDIX B

War, Siege & Conquest: The Protocol

The basic network protocol is quite simple, with a single TCP connection to the server, `mmosg.c24.finals` on port 4711. The server only responds to client requests, so the client must actively query the server for data at regular intervals. The server will respond to every client command, either with an Error, or with an OK, possibly followed by some data.

B.1 General structure

Virtually everything in the game is represented by integers. There are various types of objects, map objects (e.g. cities, troops, armies), which all have type identifiers. Almost every object also has an id number. In general, the combination (type, id) will uniquely identify an object. Id numbers *may* be reused between different types, and type numbers may be reused between different type classes (e.g. city buildings and company types).

In general, your client should, after logging in, regularly issue Poll commands which download all updates since the last Poll command. Most other commands do not give any information in return, only an OK or an Error with the error message. It is recommended that you issue the Poll command once every **2–5 seconds**.

An OK reply consists of the word OK followed by an integer denoting the number of reply lines that follow. See Section B.4.1 for information on replies.

Some information in the protocol is given as text intended for humans to read (and not to be machine parsable). These messages are sent as CHAT messages, coming from the neutral country. No CHAT messages will come from other users in this version of the server. You may wish to display these, since they can give some extra information, but that is of course optional.

All tokens in the protocol are space-separated.

Your client should end lines with just a Line Feed character. The server will also end all its lines with just a Line Feed character.

B.2 Samples

A few samples are given. Client commands look like `this` and server replies look like **this**.

B.2.1 Logging in

To log in as user **user01** with password **pass01**, the exchange might look like (with a failed password thrown in for good measure):
`USER user01`
OK 1
A
`PASS user01`
ERR BadLogin 0
`PASS pass01`
OK 0
`WORLD 1`
OK 14
STATICNAMEINFO 51 ...
STATICBUILDINGINFO 30 Academy 200 150 0 20 0 10 0 0 1 202 6 ...
STATICCOMPANYINFO 19 ...
STATICNONCOMBATINFO 3 General 11 250 0 0 0 0 0 0 6 ...
...

B.2.2 Building a building

The client decides it wishes to build a watchtower and an inn in the city Lak_Oll (cityID 133, as we will see in the CITYSTATE returned by a Poll). In the StaticBuildingInfo, we have learned that Watchtower has type 220 and that Inn has the type 208. The city already has an Inn, so that command will fail (IllegalGameObject). We also wish to construct a Pegasi Pond (type 215) in city Lum_Sal (cityID 160), but we cannot afford it, so that command also fails.
`Poll`
OK 10
...
CITYSTATE 2 133 43 5 Lak_Oll 15 ...
...
`BuildCityBuilding 133 208`
ERR IllegalGameObject 0
`BuildCityBuilding 133 220`
OK 0
`BuildCityBuilding 160 215`
ERR InsufficientResources 0

B.3 Protocol Commands

B.3.1 Login commands

User *username*
Begins login with username *username*. Reply will be OK with 1 reply line. For this task, you can ignore this line (normally part of challenge-response login). Username is case sensitive. Note that giving a non-existing username will not give an error message here, but will instead give BadLogin upon issuing the Pass command.
Pass *password*
Authenticates in with password *password*, after issuing a User command. Passwords are case sensitive. No reply data.

Poll *(no argument)*
Polls the server for queued status updates. Issue this command periodically, and rather frequently (once every 2-5 seconds is good).
Quit *(no arguments)*
Quits the game.
World *worldID*
Selects a world after logging in. This command completes the login. For this task, worldID will always be 1. Will send a big reply consisting of:

- StaticNameInfo
- SraticBuildingInfo
- StaticCompanyInfo
- StaticNonCombatInfo
- CountryState
- Mapstate
- GivenTreaties
- ReceivedTreaties
- GivenOffers
- ReceivedOffers

See Section B.4.3 for information about these messages.

B.3.2 City commands

Note that there is no way to undo a build order, so be careful when you issue them.
BuildCityBuilding *cityID, type*
Queues the construction of a building of type *type* (integer) in the city *cityID* (integer). Note that the cost of the building will be deducted from your economy immediately upon issuing the command. You must have all resources needed when you issue the command, otherwise it will fail. You must also have all prerequisites constructed.
ProduceCompany *cityID, type*
Queues the construction of a company of type *type* (integer) in the city *cityID* (integer). Note that the cost of the company will be deducted from your economy immediately upon issuing the command. You must have all resources needed when you issue the command, otherwise it will fail. You must also have all prerequisites constructed.
ProduceUnit *cityID, type*
Queues the construction of a unit (i.e. General, Engineer or Explorer) of type *type* (integer) in the city *cityID* (integer). Note that the cost of the unit will be deducted from your economy immediately upon issuing the command. You must have all resources needed when you issue the command, otherwise it will fail. You must also have all prerequisites constructed.

B.3.3 Fort commands

BuildFortBuilding *fortID, type*
Queues the construction of a building of type *type* (integer) in the fort *fortID* (integer). Note that the cost of the building will be deducted from your economy immediately upon issuing the command. You must have all resources needed when you issue the command, otherwise it will fail. You must also have all prerequisites constructed.

B.3.4 Military commands

Paths are specified as a sequence of x, y pairs. Note that the server does *not* do any path finding for you (this is a feature, not a bug. Really.). This means that paths must be specified as a sequence of adjacent (vertically, horisontally or diagonally) squares. The path *must* begin with the square you are standing on, and end with the target. Each step in the path is verified when the move is executed, so you can enter illegal paths without getting error messages. Should an error occur during execution of the path, the unit will stop.

An empty path can be specified either as a path of length zero, or a path of length two giving the current position of the unit. Path length is twice what you'd expect, i.e. for a path consisting of 7 squares, path length would be 14. Path length must always be even.

SetPath *unitID, pathLength, path*
Sets the path for unit *unitID* to the path specified by *pathLength* and *path*.

SendCcompanies *sourceType, sourceID, targetType, targetID, numberOfCompanies, companyIDs, pathLength, path*
Sends companies from source of type *sourceType* and id *sourceID* (can be a city, a fort, or an army). This is done by moving the *numberofCompanies* listed in the *companyIDs* into a troop which is given the path specified by *pathLength*, *path* and which is moving to the target of *targetType* and id *targetID*. If the target moves away from where it was when this command is given, you will probably want to give a new order (a WalkToTarget) to update the path for the troop unit. This will *not* be done automatically by the server.

WalkToTarget *sourceType, sourceID, targetType, targetID, pathLength, path*
Updates the path for a Troop walking with companies. The parameter *sourceType* must be the type of a Troop Unit. Other parameters work as in the SendCompanies command. This is useful if the target for a troop either moves or you change your mind on where to send the troop.

B.3.5 Engineer commands

BuildColony *unitID*
Tells the engineer with id *unitID* to start constructing a colony at her current location (i.e. she will not continue walking on any path before starting construction). You must have the construction costs available at the start of construction. A colony can only be built on land you own, and the map square cannot have any other buildings on it. Consumes the engineer.

BuildFort *unitID*
Tells the engineer with id *unitID* to start constructing a fort at her current location (i.e. she will not continue walking on any path before starting construction). You must have the construction costs available at the start of construction. A fort can only be built on land you own, and the map square cannot have any other buildings on it. Consumes the engineer.

B.3.6 Country commands

CountryFigures *draftRate, manaTreshold, lumberTreshold, oreTreshold, ironTreshold, ironConversion, steelConversion, mythrilConversion*
This command sets all the resource production parameters for your country. All parameters are integers. Draft rate will be capped to the range [0, 30], treshholds will be capped to the range $[0, 2^{31} - 1]$ and conversion rates will be capped by your maximum conversion rate, as determined by buildings available.

B.3.7 Diplomacy commands

MakeOffer *receiver, offeredSettings, offeredResources, demandedSettings, demandedResources, treatyTime, offerTime*
Makes a new offer to country *receiver*. Treaty settings are represented by three integers (0 (false) or 1 (true)). The first integer indicates peace, second indicates granting troop passage and third indicates granting shared vision. An offer cannot include troop passage/shared vision without including peace. Peace must be symmteric between *offeredSettings* and *demandedSettings*, but troop passage and shared vision need not be.

Resources are indicated by 7 integers, indicating the amounts of gold, mana, lumber, ore, iron, steel, and mythril (in that order).

TreatyTime is for how many gaia weeks the treaty will be in effect, should it be accepted, and *OfferTime* is for how many gaia weeks the offer is valid.
AcceptOffer *offerID*
Accepts the offer *offerID*. You must have any resources demanded available immediately.
DeclineOffer *offerID*
Declines the offer *offerID*.

B.4 Protocol replies

B.4.1 General Information

All tokens are space-separated and no tokens can contain space, except for CHAT messages (where everything until the end of the line will be the message). There are several recurring types of information.

In general, lists of things will always contain an integer first indicating how large the list is, and then all the objects in the list.

Information about your objects is generally best read via the various STATE informations (e.g. CITYSTATE and UNITSTATE). For objects belonging to other players and the neutral country, information must be read from the MAPSTATE and SQUARESTATE.

The protocol always sends new state specifications for objects when things change, rather than information about what changed. This is to make everything easier to parse.

After you issue a command, the server will reply with either OK followed by an integer indicating how many lines of reply information follow, or an ERR, followed by a string (without spaces) specifying the error type, followed by an integer listing the number of lines with reply information that follow.

B.4.2 Data types in replies

Position

Given as two integers, x and y coordinate. The world is always 256 by 256 large and it wraps around all edges (i.e. the square 0,0 is diagonally adjacent to 255,255).

CompletedLevel

The completedLevel is important for most constructable things. When something is finished, its completedLevel will be at 100. During construction, it will start out

as 0 and gradually increase to 100. An object does not really "exist" (and thus not operational) until completedLevel is 100.

Buildings

Gives a list of buildings in a fort, city or colony (colonies cannot have buildings in this version, so for those it will always be empty). Starts with an integer indicating number of buildings to follow, and then for each building the building type and completedLevel.

Resources and resource production

Resources are almost always given as a list of the 7 resources gold, mana, lumber, ore, iron, steel, and mythril in that order. This is the case even in places where some of them are constant 0, to simplify parsing.

Companies

Several objects (cities, forts, generals) can hold companies. Lists of companies are given as the number of companies and then, for each company, companyType, CompanyID, morale, battleExp, fatigue, numberofSoldiers and completedLevel. You can safely ignore the parameters morale, battleExp and fatigue. NumberofSoldiers indicates how many soldiers remain in the company (when produced, this is always 200). When in battle, companies can get partially destroyed, i.e. numberOfSoldiers can drop below 200.

EnabledProduction

In places where things can be constructed, EnabledProduction lists what can be constructed (this is determined by what prerequisites have been built). It gives the number of things which can be constructed there, and then a list of the types of the objects.

ProductionQueue

In places where things can be produced, there is generally a production queue. This lists the number of things in the queue, and the types of objects that will be produced, in the order they will be built.

AggressorRadius

All aggressive things have an aggression radius within which they are aggressive. This is an integer specifying this radius. Distances are measured in manhattan distance (i.e. $|x_1 - x_2| + |y_1 - y_2|$).

IsTroopTarget

IsTroopTarget is an integer (0 or 1) indicating wether some troop has this building/unit as its target. This can be useful since it indicates that if the unit is moved, some troop probably needs to have its path updated too.

Paths

Paths are specified by twice the length of the remaining path, and then pairs of x y coordinates for the remaining path.

General

A general leads an army. It consists of a string indicating the name, an integer giving the generals experience, and two booleans (i.e. integers which are 0 or 1). All of these can safely be ignored in this version.

MapObject

As part of the description of a Map Square there will be a list of map objects on that square. The description of a map object varies slightly by type (cities contain more information). The information starts with the type, then the id, then the countryID of the owner and the position of the object. If the type is city (type 2), then follow the city name (string, without whitespace), the city type (integer) and the city size (integer). The city type here is since a *few* neutral cities on the map are inhabited by special races. These cannot be be conquered, but when attacked will give various effects. These types are Elf (1102), Orc (1202), Dwarf (1302) and Undead (1402).

Offer

An offer is specified by the other country, the offerID, the settings and resources for the giver and the settings and resources for the receiver. Settings are specified by three 0/1 integers indicating peace, passing and share-vision. Resources are specified as usual.

Treaty

A treaty is specified with the countryID of the other country, the treaty ID, three 0/1 integers indicating peace, passing rights and shared vision, and finally an integer indicating the number of gaia weeks remaining of the treaty.

BuildingTypeInfo

Buildingtypeinfo is the static information for a building type. The first item will be a string giving the human-readable name of the building type. Then follows an integer specifying the building type. Then follows a resource specification of the building cost. Then comes a list of dependencies, which consists of the number of dependencies, and then the types of buildings which must be present before this building can be constructed. Finally, the info contains the construction time (in gaia weeks) for the building.

CompanyStats

A list of 7 integers, giving attack power, defense, magic resistance, ranged attack power, rate of fire, stamina and speed. You can safely ignore rate of fire, stamina and speed, which do not affect anything in this version.

CompanyTypeInfo

Lists information for a specific company type. Gives the human-readable name (string) and the type identifier (integer). Then follow companystats. After that comes a resource specification, giving the construction cost. After that comes the number of dependancies, and a list of all building types the company type depends on (i.e. must be present for the company to be constructable in a city). Finally comes the construction time (in gaia weeks).

UnitTypeInfo

Gives information for a unit (i.e. Explorer, Engineer and General/Army). Starts by giving a string for the human-readable form, and then an integer identifier for the type. Then comes a resource list, giving the cost to construct the unit. Finally comes the construction time.

B.4.3 Replies

CITYSTATE *type, ID, position, name, buildings, population, resourceProduction, companies, isTroopTarget, enabledProduction, productionQueue*
CITYSTATE will only be sent for cities *you* own. The parameter *type* will always be 2 (the type of a city). The position is specified as two integers, x and y. The name never contains spaces. The parameter *buildings* is a list, specified as an integer telling you how many buildings will follow, and then for each building the type of the building and how completed it is (will generally be 100). *ResourceProduction* is a set of 7 integers (gold, mana, lumber, ore, iron, steel, mythril). *Companies* gives the number of companies, and then for each company the companyType, companyID, morale, battleExp, fatigue, numberOfSoldiers and completedLevel (less than 100 if under construction). *EnabledProduction* give a list of all things (buildings, troops, units) which can be built (all prerequisites are available) in the city. It starts with an integer indicating the number of types, and then lists all types. *ProductionQueue* lists all objects in the production queue (number of objects, and then a list of types of the objects in the queue, in the order they will be produced).
COLONYSTATE *type, ID, position, popuplation, buildings, enabledProduction, productionQueue, completedLevel*
COLONYSTATE will only be sent for colonies *you* own. Type is always 3, the type for colonies. *Population, buildings, enabledBuildings,* and *productionQueue* can be safely ignored, they are not present in this version. *CompletedLevel* exists since colonies are slowly built when an engineer builds a colony, so they start out their life at CompletedLevel 0 and then slowly increase until they get to 100 at which point they start producing resources.
FORTSTATE *type, ID, position, level, agressorRadius, buildings, companies, isTroopTarget, enabledProduction, productionQueue, completedLevel*
FORTSTATE will only be sent for forts *you* own. *Level* is the level for the fort, higher levels mean the fort has larger radius and is generally better.
UNITSTATE *type, ID, position, ...*
UNITSTATE will only be sent for units *you* own. Depending on the type of the UNITSTATE, different things will follow this header information. Information on the various types will follow.
UNITSTATE *type = Army (11), ID, position, general, companies, isTroopTarget, inBattle, completedLevel, path*
UNITSTATE will only be sent for units *you* own. *InBattle* is an integer 0/1 which indicates if the army is currently engaged in battle or not.
UNITSTATE *type = Troop (12), ID, position, companies, isTroopTarget, inBattle, path*
UNITSTATE will only be sent for units *you* own.
UNITSTATE *type = Engineer (21) or Explorer (22), ID, position, path*
UNITSTATE will only be sent for units *you* own.
REMOVEPLAYEROBJ *type, id*
REMOVEPLAYEROBJ will only be sent for units *you* owned. Means that for some reason you no longer own the unit/city/fort/colony/etc. with type *type* and ID *id*.

COUNTRYSTATE *countryID, name, cities, forts, colonies, production, neighbours, resources, resourceDiff, units, draftRate, numberOfRecruits*
This is a huge status message, which is sent upon logging in. Starts with the *countryID* (integer) and human-readable name (string) for your country. Then follow city specifications for your cities (number of cities and then that many CITYSTATE:s (without the word CITYSTATE)). Then follow fort specifications for your forts (number of forts, and then that many FORTSTATE:s (without the word FORTSTATE in each)). Then analogously for your colonies.

Then come the production specification, which is the set of integers manaTreshold, lumberTreshold, oreTreshold, ironTreshold, CurrentIronConversionRate, CurrentSteelCoversionRate, CurrentMythrilConversionRate, MaxIronConversionRate, MaxSteelConversionRate, MaxMythrilConversionRate.

Then follow a neighbourlist (in the same format as for CountryNeighbours). Then follow a list of your current resources and then a list of how much resources you gain/lose each gaia week. Then follow a units list (number of units, and then UNITSTATE:s (without that word appearing)). Then two integers giving the current draft rate and the size of the resource pool.

COUNTRYPARSSTATE *resources, production, numberOfRecruits*
Gives the information for your country. *Resources* is of type resources and gives your current resourecs. *Production* is of type resources and gives your current production. *NumberOfRecruits* is the number of recruits you currently have in your recruit pool.

GAMEOVER
Means that you have lost the game, i.e. that you have no cities remaining. You can no longer issue orders and the server will disconnect you.

SQUARESTATE *numberOfObjects, objects, squareOwner*
Gives the number of objects in the square, and then a list of MapObjects. Ends with an integer giving the countryID of the square owner (will be the countryID of the neutral country for squares which are not owned by a player country).

MAPSTATE *numberOfSquares, squareList*
Gives you square states for all map squares you can see. First list the numbers of squares, and then gives x, y coordinates for the square, along with the squarestate information (i.e. numberOfObjects, objects, squareOwner).

OBSOLETESQUARE *position*
Gives the x and y coordinated for a position which you used to see but no longer can see.

COUNTRYNEIGHBOURS *neighbourlist*
Gives a list of your neighbours. Starts with the number of neighbours and then the country identifier for all neighbours you have. A neighbour is a country you have at some point had direct contact with.

GIVENOFFERS *offerlist*
This gives a list of your current given offers. Starts with an integer indiciating the number of offers, and then a list of offers.

RECEIVEDOFFERS *offerlist*
This gives a list of your current received offers. Starts with an integer indiciating the number of offers, and then a list of offers.

GIVENTREATIES *treatylist*
This give a list of your current given treaties which are in effect. Starts with an integer indicating the number of treaties, and then a list of treaties. Note that a peace treaty will occur both here and in ReceivedTreaties.

RECEIVEDTREATIES *treatylist*
This give a list of your current received treaties which are in effect. Starts with an integer indicating the number of treaties, and then a list of treaties. Note that a peace

treaty will occur both here and in ReceivedTreaties.
STATICBUILDINGINFO *buildingInfoList*
This contains static building information (i.e. information about the various building types available in the game). It starts with an integer indicating the number of building types, and then follows a list of buildingtypeinfos. Much of the information given here is also presented in the lists given in Section A.8.
STATICCOMPANYINFO
Gives static company type information (i.e. information about the company types available). It starts with an integer indicating the number of company types, and then follows a list of CompanyTypeInfo.
STATICNAMEINFO *nameInformation, neutralCountry*
This gives the player names for all countries. Starts with an integer indicating the number of countries, and then for each country gives the pair countryID (integer) and name (string, without spaces). Finally, it gives the countryID for the neutral country (i.e. the country owning all neutral cities).
STATICNONCOMBATINFO
This command is not well named, since it gives information on all units, rather than non-combat units. That means that information for armies will also be present here. Starts with an integer indicating the number of unit types (which is 3) and then come that many UnitTypeInfos.
CHAT *sender, message*
Sender gives the countryID of the sender. Here, it will always be the ID of the neutral country. This message is a bit special, in that there are no more space-separated tokens. The sender parameter will be followed by a single space, and everything after that is part of the message.

B.5 Protocol errors

When something goes wrong with a command, an error reply will be sent. There are several different types of errors. The error messages are not very consistent or good. Sorry about that.

B.5.1 The error replies

BadCommand
A catch-all for broken command syntax, non-existing commands, etc.
BadSendDestination
You tried to send companies/a troop to an object which cannot hold companies.
IllegalGameObject
This is a bit of a catch-all. You tried to do something with an object which either doesn't exist or which you don't own, or which has not finished building.
IllegalOwner
You tried to move a company which you don't own.
IllegalParameterException
Format error for integer fields.
IllegalProduction
This is caused when attempting to set a conversion rate above the maximum allowed for your country.
InsufficientRecruits
Too few recruits when attempting to build something requiring recruits to be built.
InsufficientResources
Too few resources available to complete command.

InternalError
This is most likely a bug in the server. You probably want to report this to the judges.