9th BME International 24-hour Programming Contest

Final Problem Set

2009. 05. 02.

Nav N Go

ww.challenge24.org

Electrical Engineering Students' Hungarian Association
www.eestec.hu

MŰEGYETEM 1782
Budapest University of Technology and Economics
Department of Automation and Applied Informatics
www.aut.bme.hu

Lords of thousand islands 5

1.1 Team Website 7

1.1.1 Main page 7
1.1.2 GuitarSubmissions 7
1.1.3 RobotSubmissions 7
1.1.4 Notifications 7
1.1.5 Scores 7

1.2 Technical information: 8
1.3 IRC Rules 9

1. King of the island 11

1.4 Control of the kingdom 11
1.5 Technical details 12

1.5.1 Initial parameters 12
1.5.2 Data structures 13
1.5.3 Communication basics 14
1.5.4 Basic actions 14
1.5.5 Multi round actions 17
1.5.6 Further details 17
1.5.7 Fights 19
1.5.8 Gameplay 20
1.5.9 Phases of a round 20
1.5.10 Connection to other problems 21
1.5.11 Scores 22

2 ~~300~~ 256 23

2.1.1 Soldiers 23

2.2 The thread of the game 23

2.2.1 Initialization 23
2.2.2 Steps 23
2.2.3 The end of the game 23
2.2.4 Scoring 24
2.2.5 Management 24

2.3 Drawings 25

2.3.1 The beginning 25
2.3.2 Visibility 26
2.3.3 Steps 26

3 Cheap labour 27

3.1 The tasks 27
3.2 Technical details 27
3.3 Protocol 28

3.4 Piet Concepts 29
3.4.1 Colours 29
3.4.2 Codels 29
3.4.3 Colour Blocks 29
3.4.4 Stack 30
3.4.5 Program Execution 30
3.5 Syntax Elements 30
3.5.1 Numbers 30
3.5.2 Black Blocks and Edges 30
3.5.3 White Blocks 31
3.5.4 Commands 31
3.6 Scoring 32
4 PPPoS – Pirate to Pirate Protocol over the Sea 33
4.1 Input specification 33
4.2 Important notes 33
4.3 Flags specification 33
4.4 How to send messages to a server 34
4.5 For example: 34
4.6 Scoring 35
5 Colossus of Gramming 36
5.1 Technical details 36
5.2 Your response 37
5.3 Protocol 38
6 Deathtrap Dungeon 39
6.1 Description of a maze 39
6.2 Your response 40
6.3 Protocol 40
6.4 Scoring 41
7 Master of voices 42
7.1 Technical details 42
7.2 Scoring 43
8 Sail race 44
8.1 Practice and competition mode 45
8.2 Physics 45
8.2.1 Accelerations 45
8.2.2 Calculation 46
8.3 Communication 47
8.4 Protocol description 47

8.4.1 Data types 47
8.4.2 Log in phase 48
8.4.3 Weather and track information phase 48
8.4.4 Playing phase 49
8.5 Scoring 51
9 Shopping 52
9.1 Problem specification 52
10 Treasure Island 55
10.1 Technical details: 55
10.2 Instrcutions for the robot 55
10.3 Important notes: 56
10.4 Scoring 56

Lords of thousand islands

Challenge24 2009.

Introduction

Welcome to the 9th BME International 24-hour Programming Contest's final.

You have 24 hours to solve the tasks. You can solve any of the problems you want, none of them is compulsory. Some of the tasks will be checked/verified during the competition, only task "King of the Island" will be verified after the contest.

At this year's competition we have some not-only-computer tasks ("Master of voices", "Treasure Island"). These can only be solved by one team at a time, so use your time wisely. Every team will have equal time, but at the end of the termin another party will gain access. We try to provide access to this tasks for all teams but our resources are limited. On the website you can submit a request to the organizers. After that the organizers will order all the teams according to the same way the request came in and they give you a time period in order to deal with this task. If you won't connect to client computer of the task another party with another team will start. Those teams which did not send a solution will have advantage over the teams who already did it. If you submitted a request please visit the TeamSite frequently. If you would like this task to be scored you have to notify the organizers beforehand.

You can get all togerher maximum 160.000 points after completing all tasks. For "King of the Island" you get maximum 66.000 amount of points. For the remaining tasks you can get point in the following way:

- PPPoS 10.000 points
- Deathtrap Maze 10.000 points
- Colossus of Gramming 10.000 points
- Fast and Furious 10.000 points
- Shopping 10.000 points
- Master of voices 10.000 points
- Treasure Island 10.000 points
- ~~300~~ 256 10.000 points
- Cheap Labour 14.000 points

You will get the points above if you have the maximum score (except Cheap Labour), score and point are described later.

1.1 Team Website

Every team has their own team website for the finals that you can access through the network. This site is for making your life easier during the contest. You can use the following functions:

1.1.1 Main page

You can see your team name (probably it won't be new information for you), your current IP address and your team's current ranking (if you see 1st there, you can be happy).

1.1.2 GuitarSubmissions

Here you can submit a request of submission for the "Master of Voices" problem. The organizers will receive your request and grant you a time when you can start submit for that problem. You will be able to see your submission time there (just don't forget to refresh the page)

1.1.3 RobotSubmissions

Same as for the GuitarSubmissions, except the fact that here you can request submission time for the "Treasure Island" problem.

1.1.4 Notifications

This is the place where you can read general or problem specific notifications by the organizers. For example "The food is served!" ☺

1.1.5 Scores

You can browse points and scores your team got here with some additional (probably) useful information for you.

Every solution you submit will receive a *score*. This is a number that describes the value of the solution on an absolute scale. Typically it will be calculated from the number of steps your solution needs to complete or the elapsed time. Based on this score we will rank the teams per task, and based on these rankings will you receive your *points*. The sum of the points received will determine the final ranking of the teams.

The problems can be put in different categories: for one of them we need all the teams together, while others can be submitted and verified individually.

The task of the first category will be scored the following way:

Task's name	First run	Multiplier1	Second run	Multiplier2	Third run	Multiplier3
~~300~~ 256	20:00	0.3	02:00	0.7	07:00	1

The sharp jury servers will be started the presented time. We leave 5 minutes for every team to join, and then we start the scoring. During the scoring, the computer cannot be controlled. You can only run your program. The organizers will check every team that is currently active.

At the end the final score for these problems:

```
Point = First run point * Multiplier1 + Second run point * Multiplier2 + Third run point * Multiplier3
```

The main task (“King of the Island”) will be scored after the competition.

Task from the second category (which can be checked individually) should submit till the end of the competition. List of tasks from the second category:

- PPPoS
- Cheap Labour
- Deathtrap Maze
- Colossus of Gramming
- Fast and Furious
- Shopping
- Master of voices
- Treasure Island

You have to show your intent for submit the problem, and the organizers will start a jury server. After the first run there is no way for correction. During the scoring, the computer cannot be controlled. You can only run your program. We ask you to call an organizer if you want to submit these problems and they will verify this. Otherwise the submitting is automated. We will refresh the results permanently on the TeamSite.

We ask all teams, the use the servers properly. Don’t send useless data via TCP/IP.

During the competition you can work on your own computers, use the language you want. We will announce an IP address, and a port number for each task. We will send the messages in TCP/IP packages.

For solving the problems, you can use the printed material, you had at the start of the competition, but from this point on every outdoor help is forbidden! The teams, who use any forbidden tool, will be disqualified immediately. We well check the teams several times during the competition.

1.2 Technical information:

The messages sent by the server are finished with \r\n instead if \r. We expect the data in the same form.

You will get the messages with ASCII coding. We expect them back in the same way.

The organizers will be accessible permanently via IRC.

1.3 IRC Rules

- Only conversations in English language are allowed
- The server is restricted to one channel (#challenge24), which you will automatically join on connect.
- Any kind of private messaging is prohibited and disabled
- You must not use the channel for smalltalk or any unrelated topics
- We will use the channel to announce common informations, so it is in your interest to keep an eye on it
- What nickname you must use:
    - <teamname>|<name>, where <teamname> is one of the following:
        - define, b, garbage, grotsch, ketlo, ltuni, monkey, hankey, ncu1, nohum, oerlikon, profit, procr, pudding, royal, scorp, tnu, itello, teddy, three, nintendo, sies, until, vegz, warm, qwerty, ozv
        - The name should be a derivation of your full name
- Asking questions:
- The organizers in charge of each problem have the nickname:
    - <number of problem>|nickname
- State a question in the form:
    - <number of problem>|The question it self
- Try not to ask questions if there is already an open question

There are two servers specified for each task. One of them is a test server that you can use to test your programs. You can send any number of solutions to the test servers. The other server is for the "real" submissions. When you connect to it you will start the submission process. The exact process may be different for each task, but typically you won't be able to cancel it and you can submit your solutions only once during the contest.

Good luck for solving the problems!

The Challenge24 Team

Foreword

Welcome stranger,

Please make yourself comfortable as I will tell you the legend of *The Thousand Islands…*

…at the first, there was only darkness and space. The one and only God, Twentorous Fourakiles[1] said, „Let there be light": and there was light and the light was good. Later, Twentorous has found that empty space lit by the light is not too exciting, thus he created the Planet. Originally, he wanted to create a planet of a perfect sphere, but unfortunately, he miscalculated something and the Planet became a torus. The torus Planet flying in the lit space was still boring. Twentorous decided to create water on the torus and create islands on it. Many, many islands have been created, maybe one thousand or even more. He has realized then that something is still missing. He said, „Let grass grow on the ground, and plants producing seed, and fruit-trees giving fruit, in which is their seed, after their sort: and it happened so. And Twentorous said, „Let the waters be full of fishes and the islands full of sheep", and it had been so. Finally, Twentorous realized that the world is not complete, without human beings, the beings with a soul. He had created the first men (the warrior and the worker) and had given them the ability to clone themselves. This is how the lands of thousands islands was created.

The men of thousand islands were happy and they lived without worries. The first era of the Planet began. The men have cloned themselves again and again, thus the original island became too small for them. Some of them learned how to build ships and moved to other islands. The men conquered one island after the other and mankind spread worldwide. For thousands of years, there was peace and tranquility.

The second era of the Planet began with a small aggressive group of warriors. They attacked other ships, they said that they were better than anyone else and declared that the island from where they came is only theirs. They created the Black Empire and attacked everyone from any other islands. The neighboring islands formed an alliance and striked back, but the Black Empire convinced some of them to join the Empire instead of fighting against it. For almost one hundred of years, alliances were made and broken. The Black Empire had collapsed, but the world was not as it has been before.

The third era, the era of kingdoms began only a few decades ago. Each island has created its own kingdom and created a massive fortification on the island. The kingdoms declared that the fortification was sacred by Twentorous. Since the men of the thousand islands are religious, all of them have accepted that the fortification cannot be attacked. In order to prove that the fortifications are sacred, kingdoms have created a tower (called the Chal' Lenge[2]) in the center. All of these towers are temples of Twentorous, the higher the tower is, the more power the kingdom obtains from their god. At least, men of the thousand islands believed so...

---

[1] The name Twentorous Fourakiles was too complicated for the ordinary people, thus later it has been transformed to a simplified form, Twenty Four.

[2] The name consists of the ancient word big ('Chale') and temple ('Lenge') implying that the tower is the biggest temple in the kingdom.

1. King of the island

You live on one of the most promising islands on the Planet. There are several legends about the powerful realm, about the hard-working workers and frightful warriors of the island. Most of the other islands fear the power of your island and all of them respect it. You are more than lucky to live on this island. Unfortunately, the old and well-known king of the island died unexpectedly a few days ago. The people of the island made an election and they have chosen you to be the tenth king[3] of the island. The inauguration ceremony is done, thus it is now the perfect time to prove everyone that the choice of the people was right and you are the perfect leader. You are also helped by an ancient prophecy saying that the tenth king of the island will conquer the whole world.

Will you be a good king? Do you know what to do? Can you lead your people to the greater victory?

We will see…

1.4 Control of the kingdom

At the beginning of the game, the kingdom consists of exactly one island. This island contains the central fortification and other fields. On the surface of the islands, there are resources: wood, sheep and corn. Below the surface, there are other kinds of resources: stone, iron and gem. You have to dig to find thes e resources and it is sometimes very hard to find the right area to create a mine. The sea is populated by fishes that are almost as delicious as sheep. Fish is widely used instead of sheep meat during long sailings. Resources on the surface reproduce automatically after an amount of time, but resources in the mines do not. Be wise and do not waste your metals/gems/stones!

Since you are the king of the island, every men of the realm must follow your orders even if it means certain death. You can give several orders to your men:

- Move (on islands)
- Build ship
- Gather resources (above the surface)
- Mine
- Board /unboard ship
- Sail
- Clone
- Block moving (move a group of men)
- Block ship building (build ship with a group of men)
- Block boarding (try to board / unboard to or from a ship with a group of men)

[3] Technically, the kingdom is directed by a triumvirate now because of historical reasons. Thus, bad news, you three in the team have equal power. Note that we will refer to you as 'the king', even if we want to refer to the triumvirate.

Some of the actions can be applied only by workers: warriors can only move, board/unboard, sail and clone themselves. Workers can apply all actions mentioned above.

The life in the world of thousands islands is not easy: your men need to eat every day, you may meet pirates on the sea and others may come to your island to kill your people. To avoid starving, you have to collect corn and meat (sheep or fish) with your workers continuously. To beat pirates on the sea, you need to create (clone) warriors that are also useful to prohibit foreigners of plundering your island. Note that fights are applied in an aggressive way: each time foreign people meet, they will fight automatically.

1.5 Technical details

The game takes place on a rectangular area (the world) that is divided into smaller pieces called cells. Since the world is a torus, thus the sides of the world area are connected (e.g. if you sail to a cell in the north of the world and keep sailing, you will be transferred to a cell on the south, the same applied in case of east – west movement).

The world contains the sea and several islands. Some of these islands are player islands, while others are no man's land. On each player island, there is a fortification. As mentioned in the introduction, foreign fortifications cannot be attacked or even visited.

On the surface you can find resources: wood, meat and corn. These resources reborn after you collect them. Note, however, that meat is relocated on the island every time you collect it. E.g. when you get a sheep, it disappears on the cell but at the same time, a sheep is born on other parts of the islands. You have to find where.

Below the surface, there are three other kinds of resources: stone, metal and gem. Your scientists have noticed that plants do not like earth above metal groups, thus it is not likely to have a wood or a corn field above a mine. However, since you do not get exact information where it is worth to mine, you have to try it on several cells. If you find any resource in a mine, it may be a good idea to try to dig down to different depth as well.

In order to make life a little bit easier, you get the state of your resource treasury in each round (how much resource you have grouped by resource types). Later in the text, we will refer to this treasury as the *resource table* of players.

1.5.1 Initial parameters

The world has many changing parameters, these parameters are sent to the clients when the game starts. The sequence of the parameters is exactly as defined below, the parameters are separated by the ';' character.

- The width of the world measured in terrain cells
- The height of the world measured in terrain cells
- The maximum depth of mines
- The time of ship building in rounds
- The time of boarding the ship
- The maximum speed of sailing

- The ratio of warriors dying when you are the victor (e.g. 0.6 means that you loose 60% of the number of the foreign warriors)
- The ratio of worker death, when they fight against warriors (e.g. 1.5 means that 30 warrior will kill 45 workers)
- The ratio of warrior death, when they fight against workers (e.g. 0.5 means that 30 worker will kill 15 warriors)
- The wood cost of ship building per round
- The stone cost of ship building per round
- The metal cost of ship building per round
- The gem cost of ship building per round
- The stone cost of warrior cloning
- The metal cost of warrior cloning
- The gem cost of warrior cloning
- The stone cost of worker cloning
- The metal cost of worker cloning
- The gem cost of worker cloning
- Scoring – the value of Stone
- Scoring – the value of Metal
- Scoring – the value of Gem
- Scoring – the value of Wood
- Scoring – the value of Meat
- Scoring – the value of Corn
- Scoring – the value of Worker
- Scoring – the value of Warrior
- Scoring – the value of Ship
- Maximum response time of clients in ms (the time between get actions and process actions as described later)
- Maximum number of round before game ends

1.5.2 Data structures

You can identify your men not by their name (you would not be able to remember so many names), but by their personal identification number that is an unsigned 64 bit integer. Each man has a unique ID. The ships of your kingdom are also identified by a uniqe ID. Note that the Ids are unique even among men and ships (a ship cannot have the same ID as a man).

Since players cannot create new men or new ships, the server manages the identifiers. When a new man, or ship is created, the new ID is sent back to the player. Uniqueness of IDs is ensured by the server.

In the communication we use several enumeration types. The main types are the following:

- **MessageCode**: widely used in communication to identify server and client messages. Use this enumeration to send actions, queries to the server and to process answers of the server.
- **ActionResult**: the server uses this enumeration to report the result of user actions.

- **ActivityType**: it used to show what a man (worker or warrior) is currently doing
- **CellType**: this enumeration is used to obtain the type of the cell (e.g. sea)
- **DirectionCode**: this enumeration is used in movement and navigation
- **ResourceType**: this enumeration is used to indentify the types of resources
- **ManType:** Used to handle workers and warriors in a unified way.
- The exact structure of the enumeration can be found in **Appendix A**.

1.5.3 Communication basics

Communication between the server and the clients is based on TCP/IP as described in the generic network section. Messages sent by, or received by the clients are always closed by '\r\n'. You can send multiple messages in a single network message, but do not forget the closing characters.

1.5.4 Basic actions

Once you have identified a man/ship, you can give him orders. In the following, the structure of these orders are detailed. Note that each man/ship can apply at most one of these actions in a round.

The first parameter of the commands is the MessageCode, namely the type of the command you want to give. The second parameter is an integer value, the MessageID that is an identifier generated by the user. The server does not check the uniquess of these IDs, but you can identify answers based on these IDs, thus, it is useful to use unique IDs. From the third parameter on, you must send action-specific parameters. Optional parameters (both input and output) are marked by italic fonts.

Answers for actions sent in the same round are collected by the server, thus, instead of many small messages; you get a long message of the form:

```
Answer_ActionResult|<AnswerList>
```

where answers in the answer list are separated by the pipe character ('|').

The exact meaning and syntax of commands and their answers are the following:

Movement

Both casts (i.e. workers and warriors) can apply this kind of action. In case of workers, we distinguish carry and normal mode. A worker is slower in case of carry mode (when carrying resources). The movement speed of normal mode workers and warriors are equal. The number of men on a cell is not maximized, but do not forget that foreign nations fight each other automatically if they are on the same cell.

MessageCode:	Commands_Move
Syntax:	MessageCode(int32)|MessageID(int32)|ManID(uint64)|DirectionCode(int32)
Answer:	MessageIDReference(int32):ActionResultType(int32);|

Build ship

Only applicable for workers. Building a ship needs a certain number of rounds, if you use two workers, the time is halved, etc. The time cannot be less than a round. Building can be applied only on seaside and only on your own island. Building a ship cost a certain amount of resources in each round. During building, the worker can be attacked. You can pause and continue building the ship any time.

MessageCode:	Commands_BuildShip
Syntax:	MessageCode(int32)|MessageID(int32)|ManID(uint64)
Answer:	MessageIDReference(int32):ActionResultType(int32): NewShipID(uint64)*:NewShipXPos(int32):NewShipYPos(int32)*;|

Get resource

This action is only applicable for workers on fields containing resources on the surface. The amount of gathered resource depends on the type of resource: for meat, the worker collects all available units, while for wood and corn, the amount is specified as an initial parameter. Since there can be more than one types of resource on a cell, thus, you have to specify what kind of resources your worker should collect. For food (meat and corn), the resource appears on the kingdoms resource table immediately. Wood, however, must be brought back to the center. Resources cannot be passed to other workers and a worker cannot collect resources any more if he already carries something. During resource gathering, the worker can be attacked.

MessageCode:	Commands_GetTerrainItem
Syntax:	MessageCode(int32)|MessageID(int32)|ManID(uint64)|ResourceType(int32)
Answer:	MessageIDReference(int32):ActionResultType(int32)*:GatheredAmount(int32)*;|

Mine

Only applicable for workers. Workers can mine anywhere on the islands except in the central fortification. When a worker starts to mine, you must specify how deep he should try to find the resources (you will collect some of the resources only on the selected level, not the levels above). For each level of depth, you have to spend two days (one to dig down and an other one to come back to the surface). If you come up to the surface and try again later to dig, you have to spend the same amount of time. The maximal depth of mines is specified as an initial parameter. The deeper the worker digs, the more chance he has to find something valuable. During mining, the worker can be attacked. Resources gained from the mine must be brought to the center.

MessageCode:	Commands_Mine
Syntax:	MessageCode(int32)|MessageID(int32)|ManID(uint64)|Depth(int32)
Answer:	MessageIDReference(int32):ActionResultType(int32);|

Board/Unboard

Applicable for workers and warriors. Before sailing, each man must spend certain amount of rounds to board the ship. During the activity, the worker/warrior can be attacked. You cannot board to a foreign ship.

MessageCode:	Commands_LoadToOrFromShip
Syntax:	MessageCode(int32)|MessageID(int32)|ManID(uint64)|ShipID(uint64)
Answer:	MessageIDReference(int32):ActionResultType(int32);|

Sail

Sailing with the ship, each man on the board moves with the ship. The ship can be attacked by other ships (of other players or pirates). Since ships can move a certain amount of cells each round, you have to specify how many fields you want to travel (Speed parameter). Note that you cannot turn in a round, your movement is always a straight line.

MessageCode:	Commands_Sail
Syntax:	MessageCode(int32)|MessageID(int32)|ShipID(uint64)| DirectionCode(int32)|Speed(int32)
Answer:	MessageIDReference(int32):ActionResultType(int32);|

Clone

Applicable for workers and warriors only on earth cells. Cloning may cost resources. At the end of the process, the new man is located on the same cell as the original one. During the activity, the worker/warrior can be attacked.

MessageCode:	Commands_Clone
Syntax:	MessageCode(int32)|MessageID(int32)|ManID(uint64)
Answer:	MessageIDReference(int32):ActionResultType(int32)*:NewManID(uint64)*;|

Block Movement

For block operations, you specify a terrain cell instead of specifying men on the cell directly. Movement is applied on the fastest group of men available on the cell. No action is performed on slower people. The action succeeded if there is at least one man that can be moved. The ID of the men moved is returned.

MessageCode:	BlockCommand_Move
Syntax:	MessageCode(int32)|MessageID(int32)|CellXPos(int32)|CellYPos(int32)| DirectionCode(int32)
Answer:	MessageIDReference(int32):ActionResultType(int32);*<IDList>*|
Where	<IDList> consists of Man IDs (uint64) separated by ':'

Block ship build

For block operations, you specify a terrain cell instead of specifying men on the cell directly. Workers on the selected cell try to build the ship. The action succeeded if there is at least one man that can build the ship.

MessageCode:	BlockCommand_BuildShip
Syntax:	MessageCode(int32)|MessageID(int32)|CellXPos(int32)|CellYPos(int32)
Answer:	MessageIDReference(int32):ActionResultType(int32)*:NewShipID(uint64)*;|

Block board/unboard

For block operations, you specify a terrain cell instead of specifying men on the cell directly. Men on the selected cell try to board to, or unboard from the ship. The action succeeded if there is at least one man that can board/unboard. You have to specify the ship by its ID. The ID of the men participating in boarding/unboarding is returned.

```
MessageCode:  BlockCommand_LoadUnloadShip
Syntax:       MessageCode(int32)|MessageID(int32)|CellXPos(int32)|CellYPos(int32)|
              ShipID(uint64)
Answer:       MessageIDReference(int32):ActionResultType(int32);<IDList>|

      Where   <IDList> consists of Man IDs (uint64) separated by ':'
```

1.5.5 Multi round actions

Multiround actions are special, because in this case you do not get an exact result of the command immediately, but only at the end of the action. This means for example, that in case of boarding, it is not neccessary to have a ship on the destination cell when the boarding is started, but the ship must be there when the action ends. Note that this also means that in this case, the Action_Started messages do not grant that the action will succeed.

The answer at the end of the multi round action is always the following:

```
Syntax:       ManID(uint64):MessageCode(int32);
```

Thus, you get the man performing the action and the final result of the action. Note that this answer does not contain the original message ID!

The following commands might be applied in a multi round way: move (when carrying); minde and ship load.

1.5.6 Further details

Automatic information

At the beginning of the game and at the end of each round you get automatic information from the server. The first automatic information contains much more information than the late ones, namely:

```
MessageType:  Answer_AutoInfo
Syntax: <CenterCellInfo>|<TerrainCellInfo>|<ResourceTable>|<ShipList>|<ManList>

where  <CenterCellInfo> is a list of

              XPosition(int32):YPosition(int32);

       <TerrainCellInfo> is a list of

              XPosition(int32):YPosition(int32);<ResourceItemList>#

              where <ResourceItemList> consists of items

                     ResourceType(int32):ResourceAmount(int32);

       <ResourceTable> is a list of

              ResourceType(int32):ResourceAmount(int32);

       <ShipList> is a list of

              ShipID(uint64):ShipXCoord(int32):ShipYCoord(int32);

       <ManList> is a list of

              ManID(uint64):ManXCoord(int32):ManYCoord(int32):
             ManType(int32):ActivityType(int32);
```

Thus you get a list of your center fields, a list of your normal terrain cells, the resource table of your kingdom, the list of your ships and your men.

The structure of automatic information you get at the end of each round is the following:

```
MessageType: Answer_AutoInfo
Syntax: <ResourceTable>|<DeathList>|<DestroyList>

where  <ResourceTable> is a list of

              ResourceType(int32):ResourceAmount(int32);

       <DeathList > is a list of

              ManID(uint64);

       <DestroyList> is a list of

              ShipID(uint64);
```

Deathlist contains all of your men dead in the round, while destroylist contains your ships destroyed in the round.

Information gathering

Each player can request information about cells, men or ships of the world. In case of cell queries, the amount of gained information depends on the cell: if the cell is part of the player's island, the player obtains full information; if there is a man of the player on the cell, or on one of the neighboring cells, the player obtains full information; if there is a man of the player on the neighbor of a neighboring cells, the player obtains basic information only. The content of a foreign central fortification cannot be obtained.

```
MessageCode:   Commands_TerrainInfo
Syntax:        MessageCode(int32)|Xcoord(int32):Ycoord(int32)
Answer(basic): Xcoord(int32):Ycoord(int32):CellType(int32)|
               ForeignManCount(int32):IsTherePirate (bool):ShipCount(int32)

Answer(full): Xcoord(int32):Ycoord(int32):CellType(int32)|PirateShipCount(int32)|
              <ships>|<resources>|<warriors>|<workers>

       Where  <ships> is a list consisting items of
              PlayerID(int32):ShipCount(int32);

              <resources> is a list consisting items of
              ResourceType(int32):ResourceCount(int32);

              <warriors> is a list consisting items of
              PlayerID(int32):WarriorCount(int32);


              <workers> is a list consisting items of
              PlayerID(int32):WorkerCount(int32);
```

You can send multiple queries in a single message, in this case, you have to use the ';' character as separator between the cell definitions.

In case of men / ship query, the situation is a little bit simpler. You are not allowed to get any information about a worker/warrior/ship of someone else, but you can always get information about

your own people/ships. You can use the same command for men and ships, since the ID is unique. In case of man query answers, ResourceType means the type of resource that the worker is currently carrying.

```
MessageCode:   Commands_ManOrShipInfo
Syntax:        MessageCode(int32)
Answer(Man):   ManID(uint64):CurrentXcoord(int32):Ycoord(int32):ManType(uint64):
               CurrentActivityType(int32):ResourceType(int32):ResourceAmount(int32);
Answer(Ship):  ShipID(uint64):CurrentXcoord(int32):Ycoord(int32):
               WarriorCountOnBoard(int32):WorkerCountOnBoard(int32);
```

You can send multiple queries in a single message, in this case, you have to use the ';' character as separator between the man/ship IDs.

1.5.7 Fights

If there are foreign nations on a cell, a fight is initiated. The rules are the same on continental and sea cells. The rules are applied in the following order (in a battle, not only one of these steps are applied, but usually one after the other):

1. The player with less warrior on the cell loses all of his warriors. The victor loses a certain amount of warriors (init parameter). If the number of warriors equals, both players loose all of the warriors.
2. If there are workers and foreign warriors on a cell, then workers and warriors die according to the ratio determined by the init parameter.
3. If there are workers on the cell, the player with less worker on the cell loses all of his workers. The victor loses a certain amount of workers (init parameter). If the number of workers equals, both players loose all of the workers.
4. If there are pirates on the field, they attack the men of players by following the same rules. Note that pirates do not have workers, they are always warriors only. If pirates are beaten by players and there is a victor among the players (the number of warriors does not equal), the victor gains a certain amount of treasure (resources, directly added to the resource table of the realm). The amount if affected by the number of pirates killed.

If a worker dies on an island and it is in carrying mode, then the carried resources are dropped, you can pick them up.

If there are more than two foreign nations on a cell, then the same algorithm is used by punishing (handling) each looser player equally and handling the victor the same way as mentioned above.

Pirates

On the sea, there are pirates. Pirates can be recognized (see information gathering). Pirates stay always in their ships, they do not like islands. The number of warriors on a pirate ship varies, but none of the pirate ships is stronger than the strongest ship of the players. Pirate ships carry treasure, which can be gathered when beating the pirates.

Pirates are highly aggressive; they attack all the ships that they see. Fortunately, the situation of the captains on pirate ships are usually not too stable: rebellions destroy pirate ships regularly. Still, the sea is always full of pirates.

1.5.8 Gameplay

The main steps of gameplay are the following:

1. **Server start**: The server is started, it listens to incoming connections.
2. **Registration:** The client connects to the server, it gets an answer of type `Answer_ConnectOK`. The client does not have to send anything to the server.
3. **Registration end:** The server closes the registration phase and send a `RegistrationPhaseFinished` to all registered clients.
4. **Start Game:** The server broadtcasts a `StartingGame` message to the registered clients, the message contains initial parameter information as describes earlier.
5. **Initial Info:** The server sends a `Answer_AutoInfo` messaged to the registered clients containing the initial automatic information explained earlier.
6. **Game:** The game is started, rounds are applied one after the other (see later).
7. **Game end:** The game ends if there is only one player left (except the initial phase when each player plays the game in its own universe), a certain amount of rounds ended or the timelimit reached. The server broadcasts a `StopGame` message and stops the communication.

1.5.9 Phases of a round

Each round consists of the following phases:

1. **Init round**: The player gets a `Play_InitRound` message and the sequence number of the round (the parameters are separated by '|').
2. **Get actions**: The player gets a `Play_GetActionFromPlayers` message, actions/information queries can be sent to the server. A timer is started, if the timer expiries, a `Play_ProcessActions` message is sent, the server does not accept actions anymore.
3. **Process terrain infoq queries**: Answers are sent for terrain-based information gathering requests, the type of the answer message is `Answer_InfoQuery`.
4. **Process man/ship info queries**: Answers are sent for man and ship based information gathering requests, the type of the answer message is `Answer_ManShipQuery`.
5. **Process multi round activities**: The server processes multi round activities, e.g. ship building and updates the world according to the changes. The players obtain a report about finished actions by a `Answer_MultiRoundAnswer` type message
6. **Process new activities**: Actions are being processed, no messages are sent to the player.

7. **Send activity answers**: Action answers (action results and potential output parameters) are sent. Note that these answers contain the ID of the original action message as well. The type of the messages is `Answer_ActionResult`.
8. **Process pirates**: Step the pirate ships. No messages are sent to the player.
9. **Process fights**: Fights are applied. No messages are sent to the player.
10. **Process food consumption**: For each player, the food consumption is calculated. If the player does not have enough food, the men start to die. The ratio of people dying is increasing for each round while starving. The further the men are from the center of the kingdom, the more chance they have to die.
11. **Process resource relocation & reborn**: Reproductive resources (e.g. wood after chopping) are added, relocateable resources (e.g. meat) gained by the players are relocated.
12. **Process auto information**: An automatic information package (type `Answer_AutoInfo`) is sent to the player containing information about resources, dead men and destroyed ships.
13. **Finish round**: The player gets a `Play_EndRound` message.

1.5.10 Connection to other problems

Being a good king and solving all the problems of your kingdom is rather hard. We have decided that controling your people cannot be separated from solving everyday problems. This means that you can obtain different advantages for the main problem by solving the small problems. More precisely, the following advantages are applied:

- **Shopping**: The winner team can buy ships 50% cheaper.
- **Death maze**: The winner team gets 1000 gems.
- **Colossus of Gramming**: The winner team gets 6000 stones.
- **300**: The winner team is more efficient in battles against other players, 20% less men are lost.
- **Master of voices**: The winner team gets 1000 gems.
- **Fast and furious**: The winner team can sail 50% faster.
- **Colorful messages**: The winner team battles pirates more efficiently. If they fight against pirates, they loose 33% less men.
- **Cheap labour**: The winner team gets 4000 iron.
- **Treasure Island**: The winner team gets 1000 gems.

1.5.11 Scores

Scores are based on scoring factor defined in the initial parameters. We multiple the amount of resources/men/ships by the appropriate factor and summarize it, e.g. if you have 50 stones and the stone factor is 0.2, your score will be increase by 10 points. Note that scores and points are different. Scores are used only for ranking the teams.

2 ~~300~~ 256

The main idea of this exercise is the attack or protection of a castle/ something else, so that the soldiers of the defender and attacker part are controlled by the participants

The track is a square grid, which is N square wide and M high. The attackers start on the left side, the defenders on the right side of the track. On the right there is a one grid wide important field, this is the Castle. The defender soldiers cannot enter the castle and the aim of the attackers is to send here as more soldier as they can in the less time.

2.1.1 Soldiers

The members of both sides are simple footsoldiers. They can move in every round one grid in any (8) direction. If there is an allied unit already on the traget field, the step will be invalid. If the enemy is standing on the target field, the stepping soldier will „knock down" (the unit knocked down cannot participate in the continuous game, it is not useable and does not take places ont he track). The track is visible in a 7x7 shaped square around each soldier. Neither does seethe defender and the attacker more than its soldiers. (The sizes and the start points are known.)

2.2 The thread of the game

2.2.1 Initialization

Both the attacker and the defender starts with M unit, which take exactly one column from the track. The attackers are standing in the 1st column on the left side, the defenders are standing in the (N-1)st column before the main target place.

2.2.2 Steps

Steps are coming one by one from each side. In every case the attacker side starts the game. In one round the steps have to be sent in one message. These steps will be perfomed in the sent order. Every soldier can step in 8 directions or remain in their place. If anybody tries to take an illegal step during the performance of the steps (tries to step onto the grid which is occupied by an allied soldier), the server will send a fault message about the wrong steps. The spoiled steps are punished with -1 point. The correct steps are accomplished and the enemy continues the game.

If an attacker enters the castle, it is impossible to get out, and the position of this one is not necessary to be sent, the defender will get it automatically.

The soldiers inside of the castle do not block others, every attacker can arrive to the same field.

2.2.3 The end of the game

The game will end, if:

- both sides are over 100 round
- all the soldiers of the attacker side is died
- all the alive soldiers of the attacker are inside of the castle
- all the soldiers of the defender side is died

- a soldier of the defender side steps enters the castle

2.2.4 Scoring

The attacker side

- If at least one soldier arrives into the castle, the attacker will get 200 score points as a „castle occupying bonus”
- Each soldier inside of the castle gets 2 score points in every round as a „loot bonus”
- If every soldier dies, it will make 100 bonus points as a „massacre bonus”.
- If every defender soldier dies, the attacker will get automatically the „castle occupying bonus” and the „loot bonus for each soldier for the remaining time of 60 rounds.
- (If at the attacker the number of the remaining soldiers is x and in this time there are k remaining round, then the attacker will get 100+x*k*2 more score points, if there are soldiers already in the castle-in this case the attacker gets the occupying bonus also, and the attacker will get 300+x*k*2 if there is no soldiers inside of the castle.)
- If all the alive soldiers have entered the castle, they will get the loot bonus in the remaining rounds according to the calculations above.
- If a defender enters the castle, the attacker will get the loot bonus for all the soldiers for the remaining rounds and also the occupying bonus and the massacre bonus.

The defender side

- If they can stop the attacker to enter the castle, it will take 500 score points,
- If all the attacker soldiers die, in will take 100 score points more.

Both sides:

- Every wrong step takes -1 score point.

2.2.5 Management

The teams will be in pairs randomly. Every pair has two games, the attacker and defender roles will change between them,

The game will be started by the server with a message of START (N,M) ATTACKER or DEFENDER

After it in the beginning of every round the server will send what it can see of the track, so the position of its own soldiers and if there is a visible enemy, then its position also.

The form of it will be:

```
STATUS TURN #TURNID OWN ID,ROW_INDEX1,COLUMN_INDEX1 ID,ROW_INDEX2,COLUMN_INDEX2...
ID,ROW_INDEXm,COLUMN_INDEXm ENEMY ROW_INDEX1,COLUMN_INDEX1 ROW_INDEX2,COLUMN_INDEX2
```

The solution of the steps are similar: the stepping side sends the identification of the soldiers it wants to use and sends the direction of the step either. If it does not send anything according to a soldier, that does not move.

```
STEP ID,DIR ... ID,DIR
```

If there is a false step, the server gives an error, the numbers of the false steps (its number in the list of steps, which made the error) and the dates of them.

```
ERROR STEP_ID,ID,DIR ... STEP_ID,ID,DIR
```

If the format of the command is false, the server will send the message like:

```
ERROR FORMATEXCEPTION
```

If the game ends, the server will send for both sides a message with the points they got.

```
END <REASON> <POINTS>
```

Here the REASON field can be:

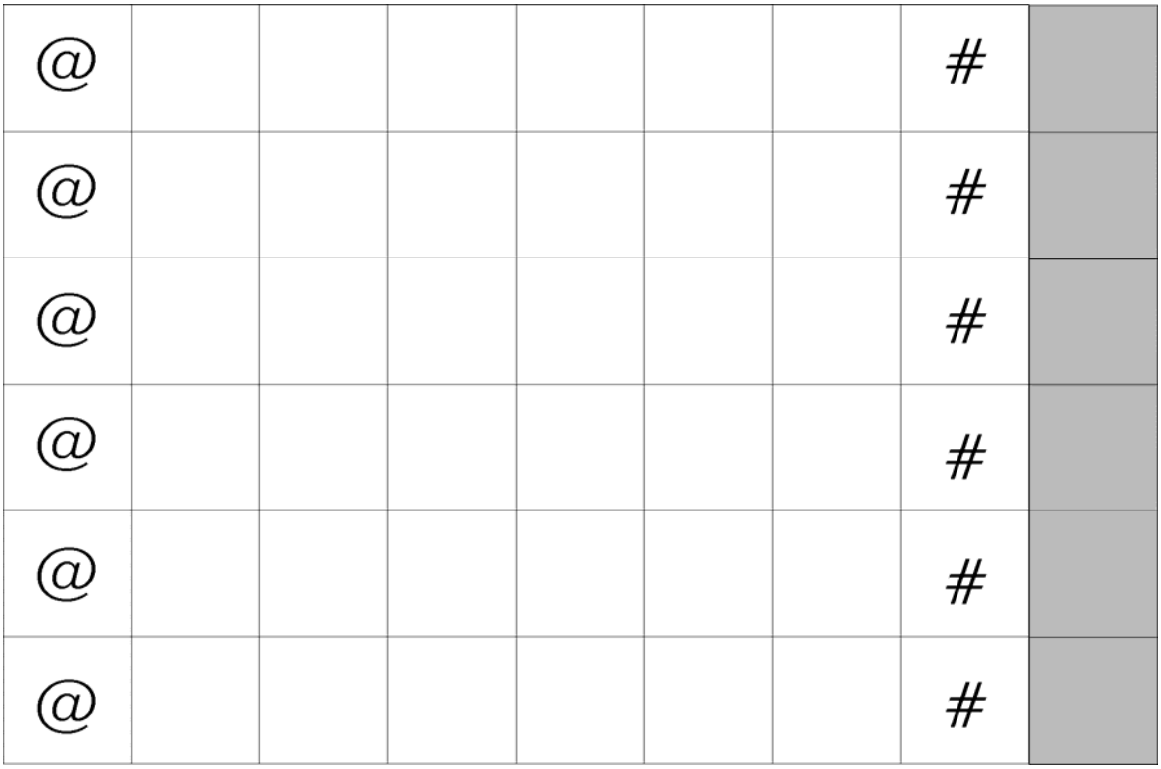
- END GAME: the 60 round limit has left.
- ATTACKER_MASSACRE: all the defenders died
- DEFENDER_MASSACRE: all the attackers died
- ALL_YOUR_BASE_ARE_BELONG_TO_US: all the soilders of the attacker entered the castle
- DEFENDER_FORFEIT: a soldier of the defender side entered the castle

The POINTS is an integer number.(POINTS>=0)

2.3 Drawings

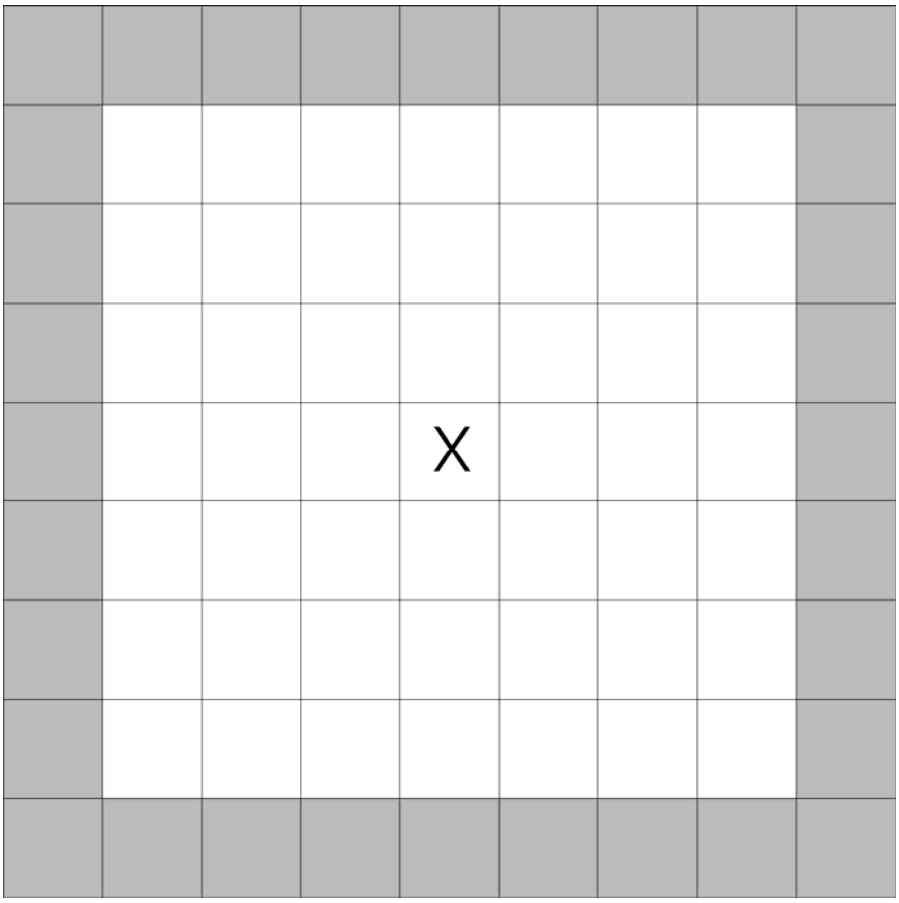
@ are the attackers, # are the defenders, X can be both.

2.3.1 The beginning



The grey grids are the castle

2.3.2 Visibility



The grey fields are already NOT visible

2.3.3 Steps

1	2	3
8	0	4
7	6	5

The soldier is standing ont he grey field, 0 means remaining (it is possible to define the remaining in explicit way, if it does not say anything, it remains)

3 Cheap labour

As a ruler of a prospering island, you have sent out many expedicions to discover the world of the Thousand Islands more. On one of these “tours” your seamen, under the command of Captain O'isskel Tachor, encountered a very friendly and very primitive tribe, the Yarman. Because of the excellent negotiating skill of Captain Tachor (and a few barrels of rum), the yarmanese have agreed to help you in your struggle to conquer the known universe. There is only one problem with this deal: the yarmanese people can’t read your written instructions, since they don’t have a usual writing. Instead, during the centuries, they have developed a unique way to preserve their knowledge (e.g. how to catch a fish). Their method is called “Piet” and it’s based on painting different colors on a piece of paper. These drawings should be read in a strictly defined way. The changing of the colors defines simple instructions (e.g. remember this number).

As the original yamanese color-coding technique is really difficult, and you will not need it, we will only give you a description of a simpler variant of it. You can find the detailed description in Appendix A.

3.1 The tasks

You currently have four tasks that you dare to trust on the yamanese. These are:

1. Write the message: `„abcdefghijklmnopqrstuvwxyz”` to the output
2. Write the message:

   `„10\r\n9\r\n8\r\n7\r\n6\r\n5\r\n4\r\n3\r\n2\r\n1\r\n”`

   to the output
3. Write the message: `„Hello world!”` to the output
4. Summing two numbers
5. Finding the greatest common divisor of two numbers
6. Finding the p-th power of an integer
7. Calculating factorial values

You have to create a “piet” drawing that describes for the yamanese the solving process of these problems. You can submit these drawings individually, they do not depend on each other. In case of each problem you must read the inputs from the standard in, and write to the standard out. In case of problems 1-3 two numbers can be read from the standard in, and a single number should be written to standard out. In case of problem 4 a single number is read and a single result is printed.

3.2 Technical details

You have to send the drawings to us in the following format:

```
P3

W H
```

```
255
R G B R G B … R G B
…
R G B R G B … R G B
```

The first line is always the characters ‘P3’. It is followd by a line describing the size of the drawing: W is the width and H is the height (two integers) separated by a single space. The next line is constant again: ‘255’. Each of the following H lines should contain exactly 3*W integers, separated by spaces. Each triplet will refer to a pixel of the drawing, specifying its color, in a standard R G B format. All values should be in the range 0<=value<=255. Lines should be terminated with \r\n.

3.3 Protocol

After you have connected, the server will send a START message. Then you can send your solution. You have two options: you can try to submit a solution or you can run tests. If you want to submit a solution, send the following message:

```
SOLUTION <id>
<drawing>
END
```

You will receive the following message:

```
SOLUTION <id>
POINTS <points>
OUTPUT
<output>
END
```

Where id specifies the problem that you have solved (values can be 1-4), drawing is the Piet code as described in Section 3.2, points is the points you have received for the solution and output is the content of the standard output that your Piet program has created.

Your other option is to run a test. The format of this is:

```
TEST <id> <input>
<drawing>
END
```

The elements of the input are separated with spaces.

As an example for the first line: “TEST 1 2 3” Id is 1, first input parameter is 2, second parameter is 3.

The response will be:

```
TEST <id>
OUTPUT
<output>
TRACE
```

```
<trace>
END
```

Where input is the content of the standard input that should be given to your program and trace is the trace of the interpreter while executing your program.

3.4 Piet Concepts

3.4.1 Colours

Piet uses 20 distinct colours, as shown in the table at right. The 18 colours in the first 3 rows of the table are related cyclically in the following two ways:

Hue Cycle: red -> yellow -> green -> cyan -> blue -> magenta -> red

Lightness Cycle: light -> normal -> dark -> light

#FFC0C0 light red	#FFFFC0 light yellow	#C0FFC0 light green	#C0FFFF light cyan	#C0C0FF light blue	#FFC0FF light magenta
#FF0000 red	#FFFF00 yellow	#00FF00 green	#00FFFF cyan	#0000FF blue	#FF00FF magenta
#C00000 dark red	#C0C000 dark yellow	#00C000 dark green	#00C0C0 dark cyan	#0000C0 dark blue	#C000C0 dark magenta
#FFFFFF white			#000000 black		

Note that "light" is considered to be one step "darker" than "dark", and vice versa. White and black do not fall into either cycle.

Additional colours (such as orange, brown) may be used, though their effect is implementation-dependent. In the simplest case, non-standard colours are treated by the language interpreter as the same as white, so may be used freely wherever white is used. (Another possibility is that they are treated the same as black.)

3.4.2 Codels

Piet code takes the form of graphics made up of the recognised colours. Individual pixels of colour are significant in the language, so it is common for programs to be enlarged for viewing so that the details are easily visible. In such enlarged programs, the term "codel" is used to mean a block of colour equivalent to a single pixel of code, to avoid confusion with the actual pixels of the enlarged graphic, of which many may make up one codel.

3.4.3 Colour Blocks

The basic unit of Piet code is the colour block. A colour block is a contiguous block of any number of codels of one colour, bounded by blocks of other colours or by the edge of the program graphic. Blocks of colour adjacent only diagonally are not considered contiguous. A colour block may be any shape and may have "holes" of other colours inside it, which are not considered part of the block.

3.4.4 Stack

Piet uses a stack for storage of all data values. Data values exist only as integers, though they may be read in or printed as Unicode character values with appropriate commands.

3.4.5 Program Execution

The Piet language interpreter begins executing a program in the colour block which includes the upper left codel of the program. The interpreter maintains a *Direction Pointer* (DP), initially pointing to the right. The DP may point either right, left, down or up. The interpreter also maintains a *Codel Chooser* (CC), initially pointing left. The CC may point either left or right. The directions of the DP and CC will often change during program execution.

As it executes the program, the interpreter traverses the colour blocks of the program under the following rules:

1. The interpreter finds the edge of the current colour block which is furthest in the direction of the DP. (This edge may be disjoint if the block is of a complex shape.)
2. The interpreter finds the codel of the current colour block on that edge which is furthest to the CC's direction of the DP's direction of travel. (Visualise this as standing on the program and walking in the direction of the DP; see table at right.)
3. The interpreter travels from that codel into the colour block containing the codel immediately in the direction of the DP.

DP	CC	Codel chosen
right	left	uppermost
	right	lowermost
down	left	rightmost
	right	leftmost
left	left	lowermost
	right	uppermost
up	left	leftmost
	right	rightmost

The interpreter continues doing this until the program terminates.

3.5 Syntax Elements

3.5.1 Numbers

Each non-black, non-white colour block in a Piet program represents an integer equal to the number of codels in that block. Note that non-positive integers cannot be represented, although they can be constructed with operators. When the interpreter encounters a number, it does not necessarily do anything with it. In particular, it is not automatically pushed on to the stack - there is an explicit command for that (see below).

3.5.2 Black Blocks and Edges

Black colour blocks and the edges of the program restrict program flow. If the Piet interpreter attempts to move into a black block or off an edge, it is stopped and the CC is toggled. The interpreter then attempts to move from its current block again. If it fails a second time, the DP is moved clockwise one step. These attempts are repeated, with the CC and DP being changed between alternate attempts. If after eight attempts the interpreter cannot leave its current colour block, there is no way out and the program terminates.

3.5.3 White Blocks

White colour blocks are "free" zones through which the interpreter passes unhindered. If it moves from a colour block into a white area, the interpreter "slides" through the white codels in the direction of the DP until it reaches a non-white colour block. If the interpreter slides into a black block or an edge, it is considered restricted (see above), otherwise it moves into the colour block so encountered. Sliding across white blocks into a new colour does not cause a command to be executed (see below). In this way, white blocks can be used to change the current colour without executing a command, which is very useful for coding loops.

- The interpreter "slides" across the white block in a straight line.
- If it hits a restriction, the CC is toggled. Since this results in no difference in where the interpreter is trying to go, the DP is immediately stepped clockwise.
- The interpreter now begins sliding from its current white codel, in the new direction of the DP, until it either enters a coloured block or encounters another restriction.
- Each time the interpreter hits a restriction while within the white block, it toggles the CC and steps the DP clockwise, then tries to slide again. This process repeats until the interpreter either enters a coloured block (where execution then continues); or until the interpreter begins retracing its route. If it retraces its route entirely within a white block, there is no way out of the white block and execution should terminate.

3.5.4 Commands

Commands are defined by the transition of colour from one colour block to the next as the interpreter travels through the program. The number of steps along the Hue Cycle and Lightness Cycle in each transition determine the command executed, as shown in the table at right. If the transition between colour blocks occurs via a slide across a white block, no command is executed. The individual commands are explained below.

	Lightness change		
Hue change	**None**	**1 Darker**	**2 Darker**
None		push	pop
1 Step	add	subtract	multiply
2 Steps	divide	mod	not
3 Steps	greater	pointer	switch
4 Steps	duplicate	roll	in(number)
5 Steps	in(char)	out(number)	out(char)

- **push:** Pushes the value of the colour block just exited on to the stack. Note that values of colour blocks are not automatically pushed on to the stack - this push operation must be explicitly carried out.
- **pop:** Pops the top value off the stack and discards it.
- **add:** Pops the top two values off the stack, adds them, and pushes the result back on the stack.
- **subtract:** Pops the top two values off the stack, subtracts the top value from the second top value, and pushes the result back on the stack.

- **multiply:** Pops the top two values off the stack, multiplies them, and pushes the result back on the stack.
- **divide:** Pops the top two values off the stack, calculates the integer division of the second top value by the top value, and pushes the result back on the stack.
- **mod:** Pops the top two values off the stack, calculates the second top value modulo the top value, and pushes the result back on the stack.
- **not:** Replaces the top value of the stack with 0 if it is non-zero, and 1 if it is zero.
- **greater:** Pops the top two values off the stack, and pushes 1 on to the stack if the second top value is greater than the top value, and pushes 0 if it is not greater.
- **pointer:** Pops the top value off the stack and rotates the DP clockwise that many steps (anticlockwise if negative).
- **switch:** Pops the top value off the stack and toggles the CC that many times.
- **duplicate:** Pushes a copy of the top value on the stack on to the stack.
- **roll:** Pops the top two values off the stack and "rolls" the remaining stack entries to a depth equal to the second value popped, by a number of rolls equal to the first value popped. A single roll to depth *n* is defined as burying the top value on the stack *n* deep and bringing all values above it up by 1 place. A negative number of rolls rolls in the opposite direction. A negative depth is an error and the command is ignored.
- **in:** Reads a value from STDIN as either a number or character, depending on the particular incarnation of this command and pushes it on to the stack.
- **out:** Pops the top value off the stack and prints it to STDOUT as either a number or character, depending on the particular incarnation of this command.

Any operations which cannot be performed (such as popping values when not enough are on the stack) are simply ignored.

3.6 Scoring

For every solved problem you will get 2000 points as a reward. Please not that this is not “score point” like in most of the other problems, but they are counted directly into the final points.

4 PPPoS – Pirate to Pirate Protocol over the Sea

Pirates are not only fearful warriors, but they are very good in ship to ship communication as well. For years, it was a great mistery how they communicate with each other without direct contact of the ships. Not too far ago, the mistery has been solved: they use color flags to send messages to other pirates' ships. Since this information is now known among the captains of your island, they have tried to use the same method. At first, it seemed working, but later they have realized that there is a large chance to miss messages because of color-blind or drunk marines who do not translate the messages as it was originally meant. The kingdom has decided to create a protocol for the communication that handles these errors and ensures trustful message passing.

Your task is to help the pirates to find out an encoding, which is always clearly identified in the message.

Exactly 50% of the flags are incorrect in the messied series.

The shorter the code is the more points you get. When you send an incorrent message back, you get penalty.

4.1 Input specification

Firstly, you have to connect to the server, which will send you a message as answer. This first message is called the original message. This is the message of your captain; you have to send it to other ships by encoding it. If you are done with the encoding, you have to send back the encoded message to the server. Note that you will receive not only one message, you have to encode and send back all of the messages in correct order.

Secondly, you must help the receiver ship. You will get encoded messages and you have to decode them. You also have to send decoded messages back to the server (to the captain) in correct order. At the end of the communication, you have to send a CLOSE command that closes the connection. Before, the communication had closed you get the score from the server.

4.2 Important notes

Each line ends with newline character. Between commands and arguments, you should use space as a separating character. Having regard to the large amounts of data, you should set the stream timeout greater than 120 seconds.

4.3 Flags specification

We have 13 kinds of flag from A to M. We specified that the flags may change to other color.

The color of the flag -> what colors to change

```
A -> G, H or D

B -> I,J,K,L or M

C -> J or I

D -> K, L or E
```

```
E -> F or B
F -> M or L
G -> M or B
H -> K or C
I -> B or A
J -> C, D or M
K -> F or A
L -> A or B
M -> H or G
```

Exactly 50% of the flags are incorrect in the messied series. For example, if my coded message is „abca", the messied message will „gbcd" or „abjh" or etc.

4.4 How to send messages to a server

So, you have the coded messages, you want to send these to the server. You have to start the message with „START" command. After that, you send your messages with „SERIES <first coded message>", „SERIES <second coded message>", etc. When the message queue is empty, you have to send a „END" command.

```
START
SERIES <first coded message>
SERIES <second coded message>
SERIES <…..>
SERIES <last coded message>
END
```

Where <… coded message> means a message encoded by you. Please send back the messages in the same order, how to get those.

4.5 For example:

Server sends:

```
START
SERIE asfsafsfsfsdfsdfsdf
SERIE asfsafsfsfsdfsdfsdf
SERIE asfsafsfsfsdfsdfsdf
END
```

Your response:

```
START
SERIE aassfffffsdfsddddfffdddssdfffaaaa
SERIE aassfffffsdfsddddfffdddssdfffaaaa
SERIE aassfffffsdfsddddfffdddssdfffaaaa
```

```
END
```

Server response:

```
START
SERIE lkssfacvbfesdasfdeasdffdssdfeaa
SERIE lkssfacvbfesdasfdeasdffdssdfeaa
SERIE lkssfacvbfesdasfdeasdffdssdfeaa
END
```

Your response:

```
START
SERIE asfsafsfsfsdfsdfsdf
SERIE asfsafsfsfsdfsdfsdf
SERIE asfsafsfsfsdfsdfsdf
END
CLOSE
```

Server sends:

```
START
POINT -1800
END
```

4.6 Scoring

You will receive score points based on the length of the tested encoded messages. Incorrectly decoded messages will receive 10000 penalty score points.

5 Colossus of Gramming

The most memorable point of your island's history was, when the old king Victorious won the Battle at Gramming against the Fellowship of The Seven Islands. Since those centuries, bards wrote new songs (were born) about The Victory, legends tell the heroism of the participating great warriors. The gripping story of your nation's persistence was told from father to son. Because your people like the kings, who remember the heroism of old times, as the new king you decided to commemorate the legendary battle. You sent runners curiers all over the world to find the best sculptor, because the Colossus of Gramming must be worthy of the ancient's glory. The old sculptor, Pr'o, who is famous for his realistic statues, proved to be the best. Unfortunately, the old master is living in on Island Sufnee, far away which is located on the other side of the world. You need a special solution because he is too old to travel and the statue is too monumental be carried on a ship. After 24 days of council your advisers elders decided: The statue will be formatted by Pr'o in Sufnee and afterwards with the aid of engineers it will be cut into pieces by engineers. The pieces will arrive on ships and the engineers will help rebuild. The challenge seemed to be hard, but it was too late to hide the plan, the people aspired after that.

Pr'o started the work and finished it in a couple of weeks. People were wondering, they sad "nobody made such a beautiful statue till this time so far". Engineers cut it into pieces with sorrowful heart and sent it home. Unfortunately at the last moment the king of Sufnee changed his mind and captured the engineers to prevent the statue being rebuilt anywhere else. He wanted the statue only hor himself.

During this, the pieces arrived by the ships but your expectation for the engineers were purposeless. With the help of your spies you got to know what happened but you could not rescue the engineers. You could only get some plans about the finished statue. People are getting unpatient day by day; they want to see the statue, whereof the legends are telling by now. This is legendary so far.

The only hope is to rebuild the statue without the engineers help. The challenge is not simple but there is no way back.

5.1 Technical details

You receive the definition of the statue and the definition of the pieces as 3D blocks. You have to build up the statue from the pieces. You must use all pieces. Although the expectations are high, you don't need to create a perfect job: it is allowed to have pieces outside the original form of the statue and you can also holes inside the statue. However, you should try to make as less errors as you can, otherwise the people will notice them.

You receive the levels and blocks in the following format:

```
LEVEL <level_id>
x,y,z
<line1>
…
<lineN>
```

```
END
BLOCKS <blocks_id>
<line1>
…
<lineM>
END
```

The first line identifies the level, then you get the x, y and z dimension of the box containing the statue. The following lines contain the definition of the level encoded by 0s and 1s. 0 means that there is space on the cell, 1 means that a part of the statue was on the cell. The level definition is followed by one END message.

After the level you will get the block definitions. The first line identifies the block. The next lines contain the definition of the blocks which is followed by one END message, each line contains exactly one block definition. Block definition consists of coordinate triples (x y z) defining the cells of the block. Block cell definitions are separated from each other by ';'. The definition 1 0 0;1 1 0;0 2 0;1 2 0;2 2 0 represents a 'T'-shape.

The <level_id> and <blocks_id> are equal for the same test case.

After you have connected you will get all the levels and blocks with the certain id-s.

5.2 Your response

You should send your solution in the following format:

```
SOLUTION <id>
<line1>
<lineD>
END
```

Where lines consist of two parts: the first part is a translation, while the second part defines a rotation, the two parts are separated by ';'. Translation is defined by x, y and z coordinates. Rotation is a list of axis – angle pairs (we use ',' between pairs and the parts of the pairs as well). For rotation axis use 0 for X, 1 for Y and 2 for Z. For angles, use 0 for 90 degree (counterclockwise), 1 for 180, 2 for 270. First, the rotation is applied then we translate the block by the translation vector.

The lines in the solution refer to the same lines in the block definition (e.g. the second line refers to the second block).

The server responses with the following message:

```
SOLUTION < id>
POINTS <points>
```

Where <id> identifies your solution and points the received score. If the score is 0, you have submitted a perfect solution. If it is -1, you have made something wrong. If the score is a positive number, then it shows how many punishment points you got. Punishment points are calculated because of holes in, or block cells outside the original statue.

You have 15 minutes to submit all your solutions.

5.3 Protocol

Once you have connected to the server, you receive a CONNECTED message. After that you will get the definition of the levels and blocks. You can solve them in any order, sending back the specified solution messages. One solution message can only contain a solution description. Every solution message is followed by a response from the server. It is either a POINTS message or a FORMAT ERROR, when you have failed to meet the specified response format.

6 Deathtrap Dungeon

Once upon a time there was an island called Braggart. The island was famous because it was so rich that even the streets were covered with gold and jewelery. The people of the island were proud and they did not fear Twentorious, instead they have sacrified for the false gods of the Karoncanat. The revenge was cruel, Twentorius released devastating earthquakes, disastrous floods that killed many of the people and destroyed their great cities. The survivors realized their fault and built a huge temple for Twentorious. Moreover, to prove their penance, they have collected all of the famous treasures, built a large labyrinth and placed the treasure inside it. When the labyrinth was ready, they killed the architects to keep its secrets. Later they built additional walls on the key paths to create a real death trap from where no man can return. The legend of Braggart is well known to every child in the Land of the Thousand Islands. Many have set out to find this miraculous treasure, but none has returned.

The crew from one of your ships has just disembarked on an unknown island and they have found strange stone walls that seem to be part of a labyrinth. They know the legends well, they know that the ancient ones have built an impossible maze, but they did not know the power of gunpowder. They have created a puzzle, which does not have any solution, but who said that you should follow their rules? The death maze may be an easy task for explosives, or at least it is worth of a try…

The task is clear: help your brave seamen to defeat the labyrinth and find the treasure.

6.1 Description of a maze

You receive a maze description in the following format:

```
MAP <map_id>
W H
StartX StartY
TreasureX TreasureY
<line1>
…
<lineH>
```

The first line identifies the map. The next line contains W and H two integers (0<=W, H<=1000), the size of the maze, width and height, respectively. StartX and StartY are the start coordinates of your men. TreasureX and TreasureY are the coordinates of the treasure (these are the coordinates you must reach).

In each of the following H lines there is a W long sequence of numbers with the following meaning:

0: Free cell. You can step on these cells without any limitations

1: Wall. You cannot step on these cells, but you can destroy them with bombs. Stepping on them is considered as an illegal move.

2: Hard wall. You cannot step on them and cannot destroy them with bombs (both are considered as an illegal move)

3: Speeder: entering this cell will increase your speed for all your following steps. Once you have entered a speeder cell, itt will change to a free cell, but you can stack multiple bonuses.

4: Slower: entering this cell will decrease your speed for all of your following steps. Once you have entered a speeder cell, itt will change to a free cell, but you can stack multiple bonuses.

Note: speeder and slower fields does not really change your movement speed (so you can still step exactly 1 cell with every command). However, they effect the scoring.

After you have connected you will get all the maps with the map id-s.

6.2 Your response

You sould send the bombs you want to place and the steps to take separately. The message format:

```
SOLUTION <map_id>
BOMBS <x1> <y1> <x2> <y2> …. <xN> <yN>
STEPS <dir1> <dir2> … <dirM>
```

<map_id> identifies the map. The following line should contain all of the bombs you wish to place, with (x, y) coordinates. Obviously 0<=x<=W, 0<=y<=H, line must contain 2*N integers if N is the number of placed bombs. If you don't want to place bombs, you should write the BOMBS word anyway, but you can leave the line empty. The numbers are separated with spaces. The next line should contain the movements of your crew. You can place the following values here:

N: go 1 cell north

S: go 1 cell south

E: go 1 cell east

W: go 1 cell west

The server should response with the following message:

```
SOLUTION <map_id>
POINTS <points>
```

Where <map_id> identifies your solution and points the received score, that is either a positive number or -1, when you have made an illegal move during your solution (e.g. stepped on a wall).

You have 15 minutes to submit all your solutions.

6.3 Protocol

Once you have connected to the server, you receive a CONNECTED message. After we send you the descriptions of the mazes, one message will contain multiple mazes. You can solve them in any order, sending back the specified solution messages. One solution message can only contain a solution description. Every solution message is followed by a response from the server. It is either a POINTS message or a FORMAT ERROR, when you have failed to meet the specified response format.

6.4 Scoring

You have to minimize your “step count” (SC). SC is increased with the current speed in case of every step you take. The speed starts from 1. Entering a speeder or slower cell will *decrease* and *increase* your speed in the following way:

```
Speeder: <new speed> = <current speed> * 0.75

Slower: <new speed> = <current speed> * 1.2
```

Additionally you receive a “bomb penalty” for every bomb you place. This increases your SC with 5.

The server success message will contain your received step count.

7 Master of voices

Each year, the islands compete with each other on a song contest called the Island Idol. The best bards of the islands sail to the beautiful Ha'why Island and try to impress the cruel and heartless jury. They say that half of the judges are already half-deaf, but still, they are well respected among the common people. The contest composes of two parts: the bards song their own composition and they prove their profiency in playing the song given by the judges.

This year, the music cuncil of your island has agreed to send the old Hoarse Pot to the contest. He is well known as the only living man on the world winning the competition twice. He is very old though, thus he was accompanied by a young singer, Cater Waul. On Ha'why everything worked perfectly: the sun shined, Hoarse Pot's performance was amazing and he won the first round, while Cater Waul got a new girlfriend. That even the stars lit a little bit brighter. From that matter, this brightness was the problem: Hoarse Pot got up in the middle of the night, he was drunk and he thought it was already daytime, thus he visited Cater Waul. He knocked on Cater's door and starts to sing. Cater Waul didn't remunerate the wake up call; they started to brawl and later to fight. Finally, Hoarse Pot fall down on the stairs and broke his right hand.

Next day, Cater Waul realized what he has done. Surprisingly, he succeeds in convincing the judges to accept him as the performer for the second part of the contest. Hoarse Pot was sulking and did not help him, thus Cater Waul has to learn how to perfectly play the guitar by himself. He has only half day which is not too much...

7.1 Technical details

You have to play the game "Guitar hero" with your program. You can do this via our server.

The server uses a binary protocol. There are two ports accepting connections:

- **4350**- this is the command port: every byte sent here is sent to the guitar controller
    - The byte sent is an OR combination of these values:
    - 4: menu down / strum bar
    - 8: 1st (green) fret button / accept
    - 16: 2nd (red) fret button / back
    - 32: 3rd (yellow) fret button
    - 64: 4th (blue) fret button
    - 128: 5th (orange) fret button
    - 1 and 2 is unused
    - The system is guaranteed to be able to press two buttons at once, but may be weak to press much more.
- **3473** - this is the video port: accepts single byte commands
    - 'a': acquire

- the server replies with a 256x192 byte grayscale, row-major image with the current camera image

Playing with the guitar controller:

1. Hold one of the fret buttons to choose note
2. Press strum bar to play note (while holding).
3. As notes cross the line, you have to play them on the guitar (a flame is shown, if the note was layed on time)

7.2 Scoring

You will receive your scores based on the game's output. This will be written into the database manually, you won't receive any automatic message about it.

8 Sail race

The pirates are holding their annual sail race to decide who going to be the fastest pirate so far at sea. There is much at stake as this title is respected and appreciated by all of the pirates. And the winner gets the opportunity to haul up a jolly roger with a flaming hair for the following year. This special Jolly Roger indicates from far that the captain of the ship is one of the greatest of the greates. Again, the stake is huge, you must win.

You have to complete a track of buoys in the shortest time possible. The wind changes with time, so you must use all your capabilities of your ship to complete the track.

Your ship is the most basic pirate ship. It doesn't have a shape of a ship; it rather looks like a big wooden bucket. Because of this, it has the same drag, whatever direction its nose is pointing to. There is a sail attached to it, perpendicularly to its nose direction (as on normal ships), and it has an anchor. Your crew can row, but that is slow way of shipping, and while rowing, you can't use the sail.

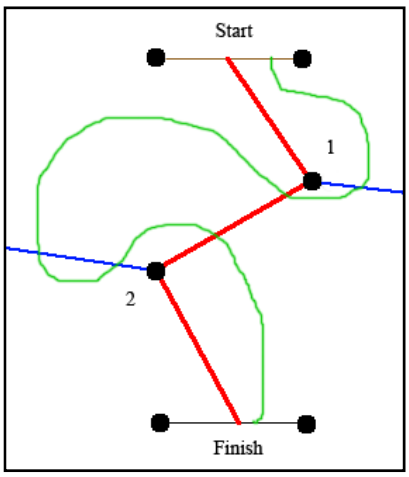
The track is made up of a start lane, a finish lane, and buoys. The start, and finish lane are specified by 2-2 points, and every buoy is specified by 1 point. The ship starts on some random point of the start lane. You must pass all the buoys on their predefined side, and pass the finish line.

For each buoy there is left or right specified, that means which side of it you must pass. Left means that looking from the previous buoy; you must pass the actual buoy on the left side. For the first buoy it means looking from the center of the starting line.

To be more precise, you must hit the bisector line (the line that divides the angle to 2 equal parts) of the course at the current buoy.

After the last buoy you can pass the finish line from any direction.

Let's see an example to clarify this:



On this track there are 2 buoys. You must pass the first buoy on its left side, and after that you must pass the second buoy on its right side. The course is shown as a red line, and the bisector at every buoy is shown with blue.
All you have to do is to first hit the blue (infinite length) line at buoy 1, and after that hit the blue line at buoy 2, and after that hit the finish line.

So if a drunk captain chooses the green path, he will compete the track, but definitely not in the shortest time.

You don't have to deal with other ships, you only have to race against the clock.
You get all the information of the track when you start a game, so you now every buoy's position in advance.

Pirates use parrots for weather forecast. They are very good at that, they can forecast the weather precisely.

You will now all the wind direction and strength in advance, when you start a game.

A game is divided to rounds. In every round, your ship's properties are sent to you, and you have to give commands to control your ship.

There is a round limit specified for every track. If you can't finish within the specified time, the game ends.

The world of the pirates lies on an infinite plane, so you can't go around, and you don't have to worry of falling off at the edge. The coordinates in the game are specified in a Cartesian coordinate system, with x coordinate increasing to the left, and y coordinate increasing upwards. There are degrees specified at some points in the game, 0 degrees means positive y direction, and 90 degrees means positive x direction.

On every track at the start, your ship is facing to the North (0˚).

8.1 Practice and competition mode

You can practice as much as you wish during the competition, and you can complete a track any number of times. When you decide to hand in the task, you can submit a track only once. Be careful, if your client disconnects, or it doesn't answer for a long period of time, your game will be ended, and you don't have more chances to complete the track.

8.2 Physics

The physics in the pirate world is almost the same as in our world. It's just a bit more discrete.

The ship is described by its position, velocity, and nose direction. In every round new velocity and position values are calculated for the ship. The simulation has nothing to deal with real clock time.

The following notation will be used:

If v is a 2d vector, then:

```
|v|: the length of v
v': is the unit vector of v
```

8.2.1 Accelerations

There are 5 acceleration factors that act on the ship:

8.2.1.1 The wind blows the ship

First a relative wind speed is calculated, that is the difference of the wind speed, and your ship's speed:

$$v_r = v_{wind} - v_{shp}$$

After this the drag is calculated is by multiplying the air resistance of your ship with the square of the length of the relative speed vector and the normalized relative speed vector:

$$a_{wind\_ship} = \rho_{wind\_ship} * |v_r|^2 * v_r'$$

8.2.1.2 The wind blows the sail

This acceleration acts in the direction of the ship's nose. First we calculate an effective speed which acts in the direction of the ship's nose:

$$v_e = d_n * (d_n * v_r)$$

Then we calculate the drag like we did before:

$a_{wind\_sail} = \rho_{wind\_sail} * |v_e|^2 * v_e'$ if the sail is up
$a_{wind\_sail} = 0$ if the sail is down

8.2.1.3 The water slows the ship

The drag is calculated from the fluid resistance and the speed of the ship:

$$a_{water\_ship} = -\rho_{water\_ship} * |v_{ship}|^2 * v_{ship}'$$

8.2.1.4 The anchor slows the ship

The anchor gives only a constant acceleration in the direction of the ship's speed:

$a_{anchor} = -c_{anchor} * v_{ship}'$ if the anchor is down
$a_{anchor} = 0$ if the anchor is up

8.2.1.5 The crew rows

This gives a constant acceleration in the direction of the nose direction:

$a_{row} = c_{row} * d_{nose}$ if rowing
$a_{row} = 0$ otherwise

8.2.2 Calculation

If the ship is moving slowly, or it has stopped, and the anchor is down, we must prevent the boat from reversing:

$$\text{if } a_{anchor} * m > v_{ship} \text{ then } v_{ship} = 0.$$

Otherwise, the final acceleration is calculated by summing the accelerations:

$$a = a_{wind\_ship} + a_{wind\_sail} + a_{water\_ship} + a_{anchor} + a_{row}$$

Then the ship's speed is calculated:

$$v_{ship} = v_{ship} + a * m$$

Then the ship's position is calculated:

$$p_{ship} = p_{ship} + v_{ship} * t$$

The values of the previous constant expressions are:

Constant	Value
ρ_{wind_ship}	0.0002
ρ_{wind_sail}	0.008
ρ_{water_ship}	20
c_{anchor}	0.05

c_{row}	0.008
t	0.1
m	30.0

8.3 Communication

The communication uses a text based protocol over TCP/IP. You have to connect to a game server to initiate a game. The practice, and competition servers are at two different locations.

Every message is sent on a new line with a terminating \r\n.

The whole communication can be divided to 3 periods. First a client has to log in, which is confirmed by the server. After this the client can request weather and track information, and in the final phase the client can start the game, and give commands to his ship.
The game at this point is round-based. The server sends a message with the current round number, and your ship's current parameters. You can give commands, or you can end the round. After this, the server steps to the next round, and sends you a message again. If your ship completes the track, or it reaches the maximal round limit of the track, the server notifies the client.

To make things clear: You are alone in the game, no others are waiting for you, so you have a relatively long time to complete a round, however, there is a timeout in the protocol.

If you don't answer to a server message in 10 minutes, the server will disconnect you. In competition mode there is also a time limit for the whole game. If you don't reach the finish line or the round limit within 10 minutes, your game will be ended.

The client can abort the session at any phase. If the client sends an unrecognizable message, or sends a message which is unexpected at the current phase, the server sends an error message.

In practice mode, a team can start any number of games, but in competition mode, only one running game per team is allowed at a time.

8.4 Protocol description

8.4.1 Data types

The specification contains some data types:

<int>: Unsigned decimal integer

<bool>: 0 or 1

<degrees>: [0..359]

<float>: A 32 bit decimal floating point number, with a '.' character for the decimal point symbol (if any).

<command_type>: [anchor, sail, row, rotate]: Types of commands you can give to your ship.

8.4.2 Log in phase

In this phase the client logs in to the server. You don't have to provide a team identifier, because it is determined by your IP address. You have to provide a track indentifier. There are 5 practice tracks, and 5 competition tracks

The client must send this message first to the server: `hello_server track=<int>`

where

`track_id` [1..5] is an identifier of the track.

If you provide a track id that does not exist, or in competition mode you try to request a track which you have already tried, the server sends you: `invalid_track.`

If you have choosen a right track, the server responds with the following message:
`hello_client player=<int>, track=<int>, round_limit=<int>`

where

`player_id` is your team id.

`track_id` is the requested track id.

`round_limit` is the maximal number of rounds, you must complete the track, or the game will end

8.4.3 Weather and track information phase

In this phase you can get weather and track information. This phase is not obligatory

8.4.3.1.1 Weather information

The weather can be requested using the following command:
`get_weather`

The server first sends the initial weather, then the changes that will happen over time.
The server sends wind change information for some specific points in time. Between these points the wind is interpolated linearly. The time of the wind information doesn't mean absolute time, but it means the number of rounds elapsed from the previous wind change. For the first wind change it means the rounds elapsed from the beginning.
After the last wind change, the first change happens again, and it goes circularly.
For example, let's consider a wind that is always blowing from the same direction, and its strength changes. Its initial strength is 0.
The wind changes that the server sends:

rounds elapse	2	3	1
strength	10	7	4

The resulting weather in the rounds:

round	0	1	**2**	3	4	**5**	**6**	7	**8**	9	10	**11**	**12**	13	**14**	**…**
strength	0	5	10	9	8	7	4	7	10	9	8	7	4	7	10	…

Linear interpolation means interpolating the winds as vectors. If two following winds are described by is w_1 in round r_1, and w_2 in round r_2, the wind w_c at the current round r_c ; ($r_1 <= r_c <= r_2$) is calculated as:

$$w_c = w_1 + (w_2 - w_1) * ((r_c - r_1) / (r_2 - r_1))$$

The initial wind information is sent using the message:

```
initial direction=<degrees>, speed=<float>
```

And the changes are sent like:

```
change elapse=<int> direction=<degrees>, speed=<float>
```

where

`round`: The number of rounds elapsing from the previous, or the initial weather change.

`direction`: The direction where the wind blows from. A wind that is blowing from 0˚ means that it is blowing to the negative y direction, and a wind that is blowing from 270˚ means that it is blowing in the positive x direction.

`speed`: The strength of the wind.

After sending all the wind changes, the server sends:
`weather_end`

8.4.3.1.2 Track information:

You can get track information by sending the message:

`get_track`

The server sends the parameters of the start line, the parameters of every buoy, and finally the finish line. The start line is sent as:

```
start_buoy index=0, position1=<float>:<float>, position2=<float>:<float>
```

where

`position1` is the x and y coordinates of one end of the start line

`position2` is the x and y coordinates of the other end of the start line

The buoys are sent as:

```
buoy index=<int>, position=<float>:<float>
```

where

`index` is the sequential number of the buoy

`position` is the x and y coordinates of the buoy.

After sending all the buoys, the server sends the finish line

```
finish_buoy index=<int>, position1=<float>:<float> position2=<float>:<float>
```

where

`index` is the sequential number of the finish line. Practically this is the number of buoys on the track. `position1` is the x and y coordinates of one end of the finish line. `position2` is the x and y coordinates of the other end of the finish line

8.4.4 Playing phase

You can start playing by sending the following command to the server:

```
start_game
```

The server replies with:

```
game_started
```

After this the server sends you the following command for every round:

```
step round=<int>, next_buoy=<int>, position=<float>:<float>, direction=<degrees>, speed=<int>, sail=<bool>, anchor=<bool>, row=<bool>, nose_direction=<degrees>, wind_direction=<degrees>, wind_speed=<float>
```

where

`round` is the number of the current round. It starts with 1.
`next_buoy` is the index of the next buoy you have to approach.
`position` is the position of your ship described with x and y coordinates.
`direction` is the heading direction of your ship.
`speed` is the speed of your ship.
`sail` describes your current sail position. 0 means your sail is down, 1 means up.
`anchor` describes your current anchor position. 0 means your anchor is down (in the depths of the sea) and 1 means up (on your deck).
`row` describes whether your crew is rowing or not.

`nose_direction` is the direction of your ship's nose is pointing to.
`wind_direction` is the direction where the wind blows from
`wind_speed` is the current speed of the wind
Now you can give commands to control your ship with the following command:
`command type=<command_type>, value=<value>`
where

`type` is the string of the type of your command
`value` is the value of your command

There are four types of commands:

- anchor: you have to specify the "anchor" string for the command_type, and the value can be 0, or 1 whether you want to raise or lower the anchor.
  0 to pull up the anchor
  1 to lower the anchor
- sail: you have to specify the "sail" string for the command_type, and the value can be 0, or 1 whether you want to raise or lower the sail.
  0 to lower the sail
  1 to raise the sail
- row: you have to specify the "row" string for the command_type, and the value can be 0, or 1 whether you want to make your crew to row or not.
  0 to stop rowing
  1 to start rowing
- rotate: you have to specify the "rotate" string for the command_type. The value specifies the new direction of the nose of your ship. The turning happens immediately

To specify more than one command, just send more of this command.
Commands that don't make any change in the ship's state are simply dropped.

Rowing and sailing at the same time is not allowed. Commands that produce this combination are simply dropped. However you can stop rowing and start sailing or vica-versa in one round, you just have to specify both commands.
It is not obligatory to send any command in a round. You can end the round by sending the following message:

```
commands_end
```

At this point if you didn't complete the track, and you are within the maximal round limit, the server sends the `step` message again.
If you just completed the track, the server sends the following message, and disconnects:

```
completed rounds=<int>
```

where

`rounds` is the overall round number of the rounds you used to complete the track

Or if you just exceeded the round limit without completing the track, the server sends the following message, and disconnects:

```
rounds_over
```

8.4.4.1 Aborting game

You can send the following message at any time to abort your connection:

```
end_game
```

8.4.4.2 Error messages

You can get these messages from the server, if some error happens:

```
time_over
```

If you exceed the time limit,, the server sends this message, and disconnects.

```
invalid_message
```

The server couldn't understand or parse your message. Or your message is not valid at the current phase of the communication.

```
game_start_is_not_allowed
```

The server sends you this message if you try to play in Contest mode, while it's not yet allowed for you to start playing.

```
invalid_track
```

You have specified a track id that does not exist.

```
invalid_team
```

There is some problem determining your team id.

8.5 Scoring

You will get higher score points if you can finish the track faster.

9 Shopping

The PillageMart is having a sale with unbeleivable prices on gun powder and firearms. No wonder all the pirates in the area want to get some. But finding a place to dock your ship can be quite challenging. Thankfully the managers of PillageMart have thought of this: a PillageMart employee fluent in flag semaphore language will watch and control the approaching ships from the control tower.

9.1 Problem specification

Your task is to help the pirate ships safely dock as soon as possible. The maximum number of steps and time is specified in each testcase. Every successful docking adds the amount of steps remaining to your score. If a ship collides with a pier or another ship, the ship or ships will stop and get stuck (meaning they cannot move any more, but they may collide with other ships). After a ship touches a trigger, the score is added, anything can happen to it: it can move, it can crash with other ships but it can't dock again.

Clients connect to the 3047 TCP port of the designated server.

After connecting, the server enters testcase selection mode and sends:

```
1 Please provide testcase number:
```

The client may now send the testcase number: "0" (or any number from 0 to 9)

If the test case is not defined, the server closes the connection.

The server now enters world mode and sends the definition of the world.

The first line contains the number of elements in the world (n).

The following n lines contain a world element definitions, which can be:

- boat <name> <x> <y> <sx> <sy> <angle>
    - name: the unique name of the boat (does not contain whitespace)
    - x, y: the x and y coordinate of the center (floating point numbers)
    - sx, sy: the width and length of the boat (boats are rectangle shaped, these are also floating point numbers)
    - angle: the angle of the boat in degrees (floating point number)
    - boats travel along their length axis (e.g. if angle is 0 they travel along the y axis)
- pier <x> <y> <sx> <sy>
    - x, y: the x and y coordinate of the center
    - sx, sy: the width and length of the pier
    - pier denotes a blocking area: if the ship touches this area, it will get stuck
- trigger <x> <y> <sx> <sy>
    - parameters are same as for pier

- trigger denotes a docking area: if the ship touches this area, it is considered docked

- steps <maxSteps>
  - maxSteps: the maximum number of steps available (the client will be disconnected after this many steps and the score will be submitted as is)
- time <maxTime>
  - maxTime: the number of seconds available (the client will be disconnected after this is over and the score will be submitted as is). This timer starts when the server receives the testcase number.

The server confirms that it is in world mode and accepts world commands by sending:

```
2 Waiting for ship commands (or end)
```

Unless otherwise noted the response for a world command may be

- <command> OK
  - sent on success
  - command: the name of the command this is the reply to
- <command> NOK [reason]
  - sent on failure
  - reason: an optional string which explains the failure (may contain whitespace, but not newline or carriage return)

The client may now send world commands:

- control <boatName> <steering> <acceleration>
  - boatName: the name of the boat to control
  - steering: the angle of the rudder (floating point, from -90 to 90)
  - acceleration: specifies how much the ship accelerates (floating point, from -4 to 4)
  - sets the state of a boat: the boat will remain in the same state, so this message may be called only when the state must be changed
  - the client may sand as many control messages as it wants, all the ships can be controlled at the same time
- step
  - steps the world
  - after ths success/failure status message the server sends the world state descriptor:
    - the first line contains the number of boats
    - the following lines contain boatState declarations: "BoatState <name> <x> <y> <angle> <crashed> <finished> <acceleration> <steering>"
    - name: the name of the boat

        - x, y: the coordinates of the center of the boat (floating)
        - angle: angle of the boat in degrees (floating)
        - crashed: 1 if the boat has crashed, 0 if not
        - finished: 1 if the boat is docked, 0 if not
        - acceleration: the current acceleration of the boat (floating)
        - steering: the current angle of the rudder in degrees (floating)
    - after the world state descriptor the server may send a "DONE <score>" message, which means the end of the simulation session
        - score: the score of the session (integer)
    - if a DONE message is sent, it is followed by a statistics block intended for human reading
        - the first line of the block is the number of lines (n)
        - the following n lines contain the statistics
        - the statistics block consist of a log of important events and final statistics
        - the actual content of the block is deliberately unspecified

10 Treasure Island

Hey stranger! Have you ever heard of the mystical lagoon of the sirens called Siren Mansion? It is said that the Chosen One who can make his way to the nest of the sirens receive far more greater reward that he ever possibly taught of: free beer as long as he lives.

Probably your first question is how can I get there? It is really simple: Give me 10 gems, and I, Larry Laffer will escort you to the entrance. After that you will have to be my eyes because I was born eons ago, so my eyesight is a bit worse than yours. I mean I can't tell the differenc between your face and a pile of apples even in the daylight.

Do you accept my offer?

10.1 Technical details:

You have to control the Lego MindStorm robot representing the player through the track and get the treasure. The robot will always start from a fixed position and the location of the treasure is the same every time, but the track might change before you start.

You can control the robot with messages sent to the server. Your software can control the robot based on the images received from two cameras, positioned near the track.

10.2 Instrcutions for the robot

You should send your instrctions to the server in the following format:

```
<motorport> <motorpower> <rotation>
```

<motorport> represents the motor you want to turn. It must be either A or B or C. A is the motor that operates the claws, B is the motor of the left wheel, and C is the motor of the right wheel.

<motorpower> is the torque of the actual motor. It is a signed integer between -100 and 100. Negative amount of whirl emphasis cause the motor to turn backwards. For the grab motor (motor A) power can only be set in the range of [-30, 30] as more power would damage the robot.

<rotation> is the rotation of the actual motor in degrees. It cannot be less than zero.

You can specifiy multiple instructions in a single message. Every instruction must end with characters \r\n.

For example:

A 30 30 opens the claws

A -30 30 closes the claws

B 25 360 turns the robot right as the B (left) motor turns forward 360 degrees and the other wheel motor does not do anything

C -20 90 turns the robot right as the C (right) motor turns backward 90 degrees and the other wheel motor does not do anything

10.3 Important notes:

The commands carried out with delay, as the controlling unit adds some latency – the server has its own delay and the robot itself is relatively slow. The robot can store only a limited number of instructions. If you send too many, it is possible that not all of them will be carried out.

The robot functions reliable within the [-30;30] Motorpower interval. For grabbing you cannot set

While a motor is performing a command, and gets a command with an opposite signed Motorpower the motor behave undifined. So don't send instructions like:

```
B -24 3000
B 21 400
```

10.4 Scoring

You will receive scores based on the time it takes for you to solve the task. This will be measured manually by the organizers and you will receive no automated response containing your score.

Appendix A

Integer values are continuous, thus for example the code of StartingGame is 3.

```
MessageCode:
        Initializing = 0,
        ListeningToRegistrationRequests,
        RegistrationPhaseFinished,
        StartingGame,
        Play_InitRound,
        Play_GetActionFromPlayers,
        Play_ProcessActions,
        Play_SendAnswers,
        Play_EndRound,
        ProcessingReconnectedClients,
        StopGame,
        Answer_ConnectOK,
        Answer_ReconnectOKWaitForNewRound,
        Answer_InfoQuery,
        Answer_ManShipQuery,
        Answer_ActionResult,
        Answer_AutoInfo,
        Answer_MultiRoundAnswer,
        Commands_TerrainInfo,
        Commands_ManOrShipInfo,
        Commands_Move,
        Commands_BuildShip,
        Commands_GetTerrainItem,
        Commands_Mine,
        Commands_LoadToOrFromShip,
        Commands_Clone,
        Commands_Sail,
        BlockCommand_Move,
        BlockCommand_LoadUnloadShip,
        BlockCommand_BuildShip

ActionResult :
        ActionSucceeded = 0,
        ManNotFound,
        CellNotFound,
```

```
        ActionNotAllowedForThisManType,
        ShipNotFound,
        CannotBoardToSea,
        ManIsInactive,
        Move_InvalidDestination,
        ShipBuild_CannotBuildOnForeignIsland,
        ShipBuild_BuildInProgress,
        ShipBuild_ShipReady,
        ShipBuild_ShipLostInDesert,
        ShipBuild_NotEnoughResource,
        Mine_InvalidDepth,
        NoTerrainItemOnCell,
        ShipLoad_CannotLoadForeignShip,
        ShipLoad_ShipIsFull,
        Clone_CannotCloneOnSea,
        Clone_NotEnoughResource,
        Sail_NotSeaCell,
        Sail_AlreadySailing,
        WorkerIsFull,
        GenericError,
        SecurityException,
        ActionStarted,
        MultiRound_ManRelocated,
        MultiRound_MineComplete,
        MultiRound_Boarded
ActivityType:
        None=0,
        Move,
        BuildShip,
        GetTerrainItem,
        Mine,
        ShipLoad,
        Sail,
        Clone
CellType:
       Sea = 0,
        Earth,
        Center
```

```
DirectionCode:
        North = 0,
        East,
        South,
        West


ResourceType:
        Stone = 0,
        Metal,
        Gem,
        Wood,
        Meat,
        Corn
```