Courses File

The data about the courses (for the server use) are given by a text file which you will define beforehand. The file name **MUST** be **Courses.txt**, and it should be located in the main folder (the project folder). The file consists of lines, where every line refers to a specific course. The format of a single line will be as follows:

courseNum|courseName|KdamCoursesList|numOfMaxStudents

courseNum: the number of the course (100>= int >= 0)

courseName: the course name (non-empty string)

KdamCoursesList: the list of the Kdam courses. (format: **[course1Num, course2Num,...]**)

numOfMaxStudents: the maximum number of students allowed to register to this course(int >= 5)

No spaces before and after the **'|'** character - you don't need to check that.

The course name doesn't contain the **'|'** character - you also don't need to check that.

You can assume that the courses file is formatted well (means, all courses lines are written in the correct format), so no need to check that.

Example:

42|How to Train Your Dragon|[43,2,32,39]|25

1.3 Supported Commands

The BGRS protocol is composed of two types of commands, Server-to-Client and Client-to-Server. The commands begin with 2 bytes (short) to describe the **opcode** . The rest of the message will be defined specifically for each command as such:

2 bytes	Length defined by command
Opcode	…

We supplied functions that encode \ decode between 2 bytes and short for both java and C++ in the assignment page.

The BGRS protocol supports 11 types of messages:

- 1)Client-to-Server messages
- 2)Server-to-Client messages

Opcode	**Operation**
1	Admin register (ADMINREG)
2	Student register (STUDENTREG)
3	Login request (LOGIN)
4	Logout request (LOGOUT)

5	Register to course (COURSEREG)
6	Check Kdam course (KDAMCHECK)
7	(Admin)Print course status (COURSESTAT)
8	(Admin)Print student status (STUDENTSTAT)
9	check if registered (ISREGISTERED)
10	Unregister to course (UNREGISTER)
11	Check my current courses (MYCOURSES)
12	Acknowledgement (ACK)
13	Error (ERR)

ADMINREG Messages:

Messages have the following format:

2 bytes	string	1 byte	string	1 byte
Opcode	Username	0	Password	0

Messages that appear only in a Client-to-Server communication.
An ADMINREG message is used to register an admin in the service. If the username is already registered in the server, an ERROR message is returned. If successful an ACK message will be sent in return. Both string parameters are a sequence of bytes in UTF-8 terminated by a zero byte (also known as the '\0' char).

Parameters:

- Opcode: 1.
- Username**:** The username to register in the server.
- Password: The password for the current username (used to log in to the server).

Command initiation:

- This command is initiated by entering the following text in the client command line interface: **ADMINREG <Username> <Password>**

STUDENTREG Messages:

Messages have the following format:

2 bytes	string	1 byte	string	1 byte
Opcode	Username	0	Password	0

Messages that appear only in a Client-to-Server communication.
A STUDENTREG message is used to register a student in the service. If the username is already registered in the server, an ERROR message is returned. If successful an ACK message will be sent in return. Both string parameters are a sequence of bytes in UTF-8 terminated by a zero byte (also known as the '\0' char).

Parameters:

- Opcode: 2.
- Username: The username to register in the server.
- Password: The password for the current username (used to log in to the server).

Command initiation:

- This command is initiated by entering the following text in the client command line interface: **STUDENTREG <Username> <Password>**

LOGIN Messages:

Messages have the following format:

2 bytes	string	1 byte	string	1 byte
Opcode	Username	0	Password	0

Messages that appear only in a Client-to-Server communication.
A LOGIN message is used to login a user into the server. If the user doesn't exist or the password doesn't match the one entered for the username, sends an ERROR message. An ERROR message should also appear if the current client has already successfully logged in. Both string parameters are a sequence of bytes in UTF-8 terminated by a zero byte.

Parameters:

- Opcode: 3.
- Username: The username to log in the server.
- Password: The password for the current username (used to log in to the server)

Command initiation:

- This command is initiated by entering the following text in the client command line interface: **LOGIN <Username> <Password>**

LOGOUT Messages:

Messages have the following format:

2 bytes
Opcode

Messages that appear only in a Client-to-Server communication. Informs the server on client disconnection. Client may terminate only after receiving an ACK message in replay. If no user is logged in, sends an ERROR message.

Parameters:

- Opcode: 4.

Command initiation:

- This command is initiated by entering the following text in the client command line interface: **LOGOUT**
- Once the **ACK** command is received in the client, it must terminate itself.

COURSEREG Messages:

Messages have the following format:

2 bytes	2 bytes
Opcode	Course Number

Messages that appear only in a Client-to-Server communication. Inform the server about the course the student want to register to, if the registration done successfully, an ACK message will be sent back to the client, otherwise, (e.g. no such course is exist, no seats are available in this course, the student does not have all the Kdam courses, the student is not logged in) ERR message will be sent back. (Note: the admin can’t register to courses, in case the admin sends a COURSEREG message, and ERR message will be sent back to the client).

Parameters

Opcode: 5

Course Number: the number of the course the student wants to register to.

Command initiation:

- This command is initiated by entering the following text in the client command line interface: **COURSEREG <CourseNum>**

KDAMCHECK Messages:

Messages have the following format:

2 bytes	2 bytes
Opcode	Course Number

Messages that appear only in a Client-to-Server communication.
KDAMCHECK this message checks what are the KDAM courses of the specified course.
If student registered to a course successfully, we consider him having this course as KDAM

Parameters:

- Opcode: 6.
- Course Number: the number of the course the user needs to know its KDAM courses

When the server gets the message it returns the list of the KDAM courses, in the SAME ORDER as in the courses file (if there are now KDAM courses it returns empty string).

Command initiation:

- This command is initiated by entering the following texts in the client command line interface:

KDAMCHECK <CourseNumber>

COURSESTAT Messages:

Messages have the following format:
(Admin Message)

2 bytes	2 bytes
Opcode	Course Number

Messages that appear only in a Client-to-Server communication.
The admin sends this message to the server to get the state of a specific course.
the client should prints the state of the course as followed:
Course: (<courseNum>) <courseName>

Seats Available: <numOfSeatsAvailable> / <maxNumOfSeats>
Students Registered: <listOfStudents> //ordered alphabetically
Example:
Course: (42) How To Train Your Dragon
Seats Available: 22/25
Students Registered: [ahufferson, hhhaddock, thevast] //if there are no students registered yet, simply print []

Parameters:

- Opcode: 7

Course Number: the number of the course we want the state of.

Command initiation:

- This command is initiated by entering the following texts in the client command line interface:

COURSESTAT <courseNum>

STUDENTSTAT Messages:

(Admin Message)
Messages have the following format:

2 bytes	String	1 byte
Opcode	Student Username	0

Messages that appear only in a Client-to-Server
A STUDENTSTAT message is used to receive a status about a specific student.
the client should print the state of the course as followed:
Student: <studentUsername>
Courses: <listOfCoursesNumbersStudentRegisteredTo> //ordered in the same order as in the courses file
Example:
Student: hhhaddock
Courses: [42] // if the student hasn't registered to any course yet, simply print []

Parameters:

- Opcode: 8.

Command initiation:

- This command is initiated by entering the following texts in the client command line interface: **STUDENTSTAT <StudentUsername>**

ISREGISTERED Messages:

Messages have the following format:

2 bytes	2 bytes
Opcode	Course Number

Messages that appear only in a Client-to-Server communication.
An ISREGISTERED message is used to know if the student is registered to the specified course.
The server send back "REGISTERED" if the student is already registered to the course, otherwise, it sends back "NOT REGISTERED".

Parameters:

- Opcode: 9.
- Course Number**:** The number of the course the student wants to check.

Command initiation:

- This command is initiated by entering the following texts in the client command line interface: **ISREGISTERED <courseNum>**

UNREGISTER Messages:

Messages have the following format:

2 bytes	2 bytes
Opcode	Course Number

Messages that appear only in a Client-to-Server communication.
An UNREGISTER message is used to unregister to a specific course
The server sends back an ACK message if the registration process successfully done, otherwise, it sends back an ERR message.

Parameters:

- Opcode: 10.
- Course Number**:** The number of the course the student wants to unregister to.

Command initiation:

- This command is initiated by entering the following texts in the client command line interface: **UNREGISTER <courseNum>**

MYCOURSES Messages:

Messages have the following format:

2 bytes
Opcode

Messages that appear only in a Client-to-Server communication.
A MYCOURSES message is used to know the courses the student has registered to.
The server sends back a list of the courses number(in the format:**[<coursenum1>,<coursenum2>]**) that the student has registered to (could be empty **[]**).

Parameters:

- Opcode: 11.

Command initiation:

- This command is initiated by entering the following texts in the client command line interface: **MYCOURSES**

-

ACK Messages:

Messages have the following format:

2 bytes	2 bytes	-	1 byte
Opcode	Message Opcode	String to be printed at the client side (as bytes)	0

Messages that appear only in a Server-to-Client communication.
ACK Messages are used to acknowledge different Messages. Each ACK contains the message number for which the ack was sent. In the optional section there will be additional data for some of the Messages (if a message uses the optional section it will be specified under the message description).
All Messages that appear in a Client-to-Server communication require an ack/error message in response.

Parameters:

- Opcode: 12.
- Message Opcode: The message opcode the ACK was sent for.
- Optional: changes for each message.

Client screen output:

- Any ACK message received in **client** should be written to the screen in the following manner:

ACK <Message Opcode>

<Optional>

- Printing of the optional part: split between optional parameters by space.

ERROR Messages:

Messages have the following format:

2 bytes	2 bytes
Opcode	Message Opcode

Messages that appear only in a Server-to-Client communication.
An ERROR message may be the acknowledgment of any other type of message. In case of error, an error message should be sent.

Parameters:

- Opcode: 13.
- Message Opcode: The message opcode the ERROR was sent for.

Error Notification:

- Any error message received in **client** should be written to screen:

ERROR <Message Opcode>

In any case, if the message cannot be processed successfully, or the user is not allowed to send this message (e.g. student sends an admin message), the server returns an ERROR message.