Information Technology. Name: _____

Grade 12 Performance Assessment Task (PAT)

For the PAT you need 4 sets of documents – 4 separate word documents:

- Project specifications
- Design Document
- Technical document
- Testing document

The documentation is VERY important - it makes up about 55% of the mark. Start with the project specifications, followed by the design document.

Create a simple Database (normalised) with at least 3 tables. You have to use SQL to insert, delete and update the tables through the GUI. No data aware components allowed, must use OOP principles.

Example:
DVD shop

tblDVD -DVDID -Name -genre etc....	tblRentals -RetalID -Date -DVDID -CLid	tblClients -CLid -ClName -ClTel -Address Etc....
DVDClass	RentalClass	ClientClass
DVDArrayClass	RentalArrayClass	ClientArrayClass
There will thus be a minimum of 6 classes. One object class and one array of object class for each table in the DB.		

PERFORMANCE ASSESSMENT TASK - PAT

Programming Project

This programming project represents the development cycle of a product. The purpose is to give the learner a meaningful experience of a larger project and development cycle than is usually not possible in a classroom situation.

Scope

This project should be more complex and include more features that those of a practical examination paper. Inappropriate examples might include a calculator or a currency converter.

The program needs to meet the following criteria:

· **User friendly -**

- **Interface:** the interface must be user friendly (preferably a GUI), easy to use and task appropriate.

- **Data Flow/program operation:** the learner must ensure that the sequence of steps required to use the program and complete a task are clear, easy to follow and logical.

Storage/Data persistence – data must be stored and retrieved from session to session (this can be in the form of conventional files OR a database OR both). Database work is to be encouraged, as is the use of SQL. The storage is appropriate to the program.

Separation of interface and engine – the 'working code' **must not be embedded in the interface** (i.e. it must be in separate classes/units). Communication between the interface and the working code is in the form of parameters and typed methods (functions). A limited amount of code in the interface is acceptable only if suitably justified in the planning.

Good data internal structures – There has to be some form of internal representation of data (i.e. classes/records/arrays – or any combination of these). Data structures must be logical and task appropriate. Classes are to be encouraged.

Phase		Description	Marks
What must be submitted by the candidate:	Project Specifications	List (and describe) the functions that your program needs to achieve in order to be a 'success'.	14
	Design Document	Design the user interface, sequencing (data flow), class and persistent storage of the program in detail.	31
	Coding and Technical Document	Write the program following good programming techniques and document it by printing the code and explaining critical algorithms.	50
	Testing Document	Document what is to be tested, the test data used and the results of the testing.	5

Example: A program to manage a school.

The four word documents that accompany your code are outlined in greater detail here with examples taken from a number of different sources.

Document 1 : Project Specification Document
Overview of your project. Summary of the problem your software will attempt to address, a list of features, the data your program will store and description of the hardware and software requirements. This section details **what** your program will do, not how it will do it.

1.1 Cover

- Performance Assessment Task
- Project Specification Document
- Your full name
- Your grade
- The year.

1.2 Table of contents

1.3 Summary
Summary of the problem your software will attempt to address, a list of features, the data your program will store and description of the hardware and software requirements. This section details **what** your program will do, not how it will do it.

1.4 Specifications of Program Function
Full and comprehensive list of the features and functions that your program will have; suggested format - bullets. Each bulleted point will turn into something that you need to create or code.

Some ideas for the school management program

- Opening screen – may have the date and who is the duty teacher
- Login screen – allows the administrator to log into the confidential information
- Display all the information of a particular student
- Add a new student into the database
- Delete a student from the database – also the mother from the mother table and the father from the father table according to the unique ID
- Update a student's details
- Search for a student using only part of their name
- Write a student's details to a text file that can be printed out.
- Search for students in a particular grade
- View the student's parents ie mother and or father
- View the student's register teacher
- Help. Displays information about the mail GUI which is read in from a text file

1.5 Specifications of the Graphic User Interface

Opening GUI

- Login – requires name and password – goes to Main GUI
- Exit button closes the program

Main GUI – some ideas that would be appropriate

- New student
  - Name, phone number, address, medical issues, grade, house, fathers name, mothers name, email, register teacher etc
  - Update button writes all data to the database
  - Clear button clears all fields
  - Back button goes back to the Main GUI
- Update student – includes information about parents
  - Update button writes to the database
  - Clear button clears all the fields
  - Back button goes back to the Main GUI
- Delete student
- View all students
- Search for student
  - Search by name
  - Search by grade
  - Search by ID
  - Back button goes back to Main GUI
- Search for parent
  - Search by name
  - Back button goes back to Main GUI
- Help button contains information on the functionality of all the buttons

1.6 Specification of Help

Program is user friendly and supplies error or success message on completion

- Login. If username and password is incorrect an error message is displayed.
- Help button on the Main GUI explaining the purposes of the button (taken from a text file)
- New Student. Error messages if any of the fields do not have data. Success notice will be displayed on successful addition

1.7 Specification of Data Storage Tables in the Access Database

Student – Table – some ideas

Field with example	Datatype
ID	AutoNumber
First Name. Eg "John"	String
Last Name. Eg "Edwards"	String
Address1. Eg "12 Ruby Rd	String
Address2 Eg "Sunningdale"	String
Address3 Eg "Randburg"	String
Country Eg "South Africa	String
Postal Code. Eg "1234"	String
Grade. Eg "12"	Integer
Mobile. Eg "0827867654"	String
Mother. Eg "Mary Edwards"	String
Father . Eg "Peter Jackson"	String
House. Eg "Saturn"	String
Email. Eg "john345@gmail.com"	String

Father – Table

etc

Mother – Table

etc

Register Teacher – Table

etc

Text File Storage

Help – Contains the text for the Help Screen

1.8) Hardware and Software Specifications

Hardware

Processor?
RAM needed?
Available space on HDD?

Software

Operating system
Java Runtime Environment?
Microsoft Access?
ODBC driver?

Document 2 : System Design Document
Details the actual design of your program i.e. GUI design, program flow, class design, database design, storage design. This will be the longest and the most detailed document. Many of the pages will contain screen shots, diagrams and images. This document details how your program is going to work.

2.1 Cover

- Performance Assessment Task
- System Design Document
- Your full name
- Your grade
- The year.

2.2 Table of contents

2.3 User Interface Design
A screen shot of every one of your GUIs and how they will work. Also how the user will interact with the various screens in your program.

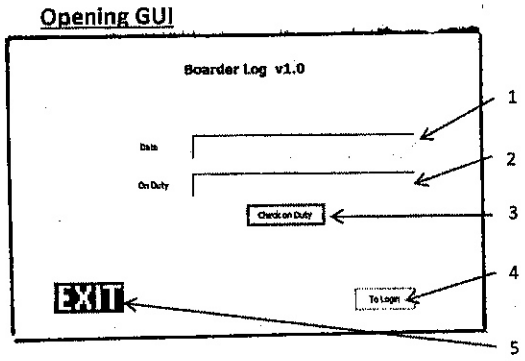
- Data to be displayed/entered on the screen
- Security (which users can access the screen)
- Action Elements (what can be clicked and what does clicking on the GUI component actually do?)

Readable fonts, good layout, suitable colours, easy-to-use input boxes, visibility of features and functions are all important.

Turn overleaf for an example of what this page could look like . . .

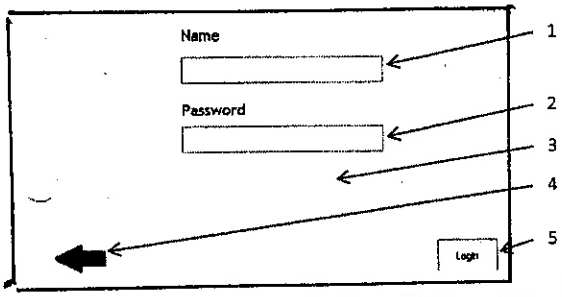
1. **USER INTERFACE DESIGN**

Opening GUI



Description	Displays the date and faculty member on duty.
Security Group	All users
Data	1. Displays the date 2. Displays the Faculty member on duty and their contact (phone) number.
Actions	3. Checks/returns the date and returns the faculty member that is on duty. 4. Moves to the login GUI. 5. Closes the program.

Login GUI

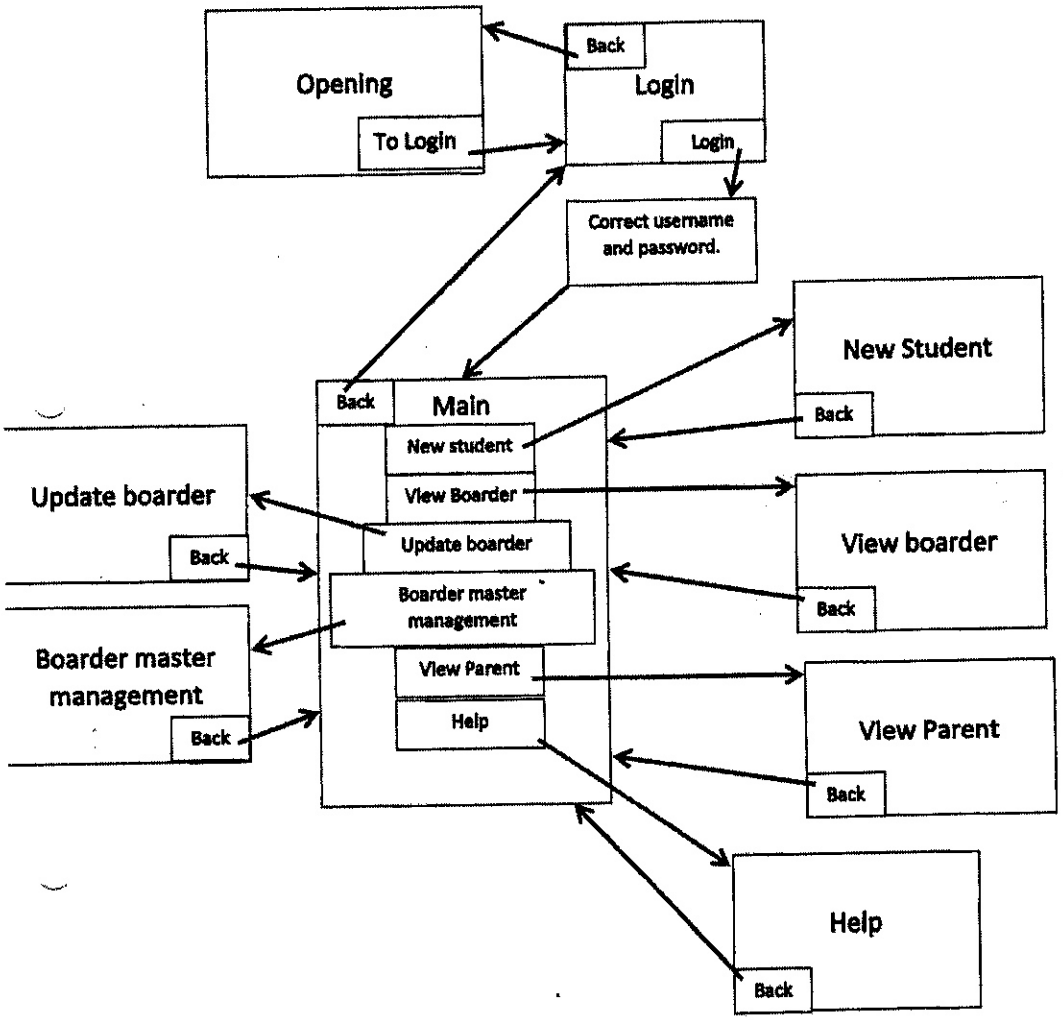


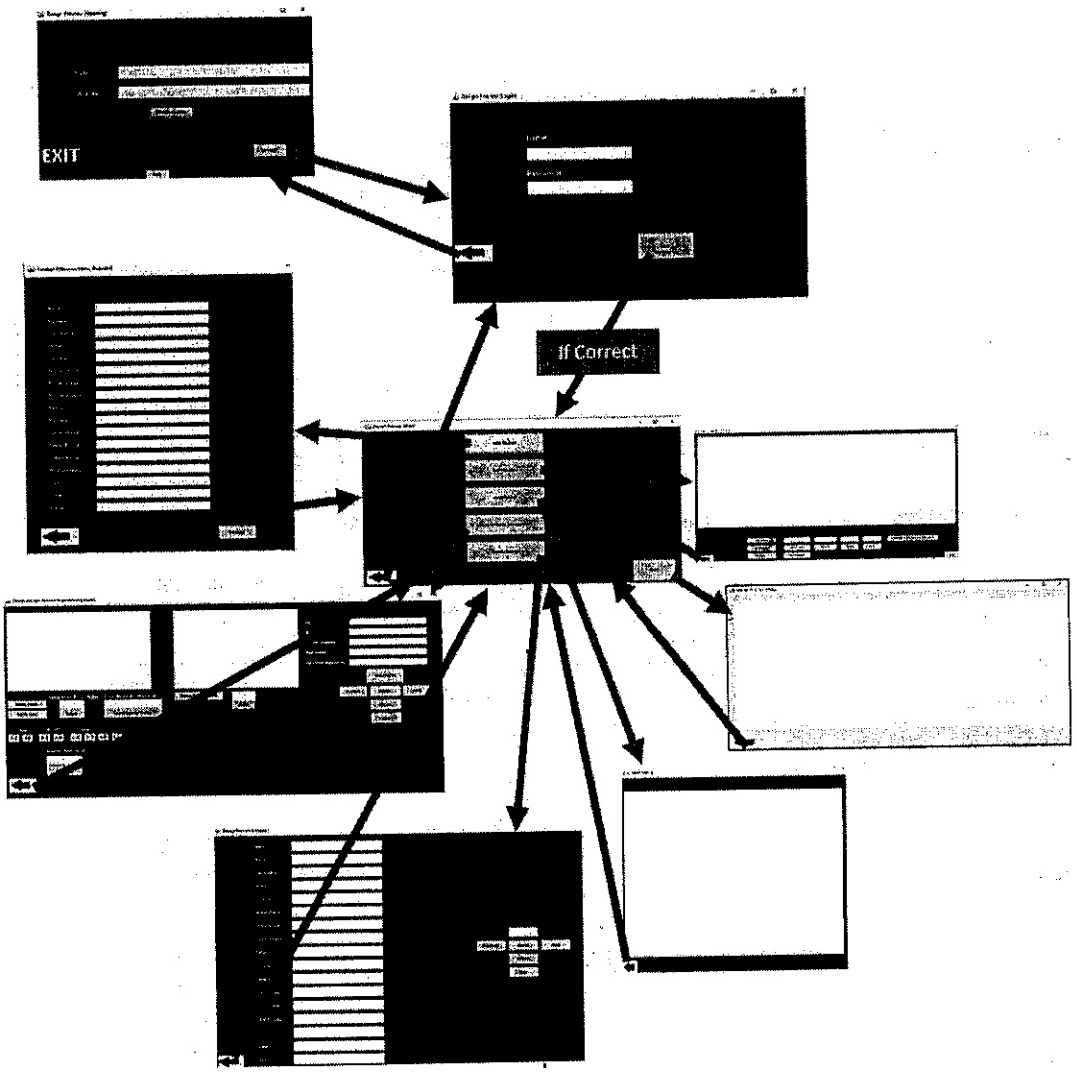
Description	User enters their username and password, program checks if they match any in the database.
Security Group	All Users
Data	1. User enters their username. 2. User enters their password. 3. Displays Error message if incorrect username or password is entered.
Actions	4. Opens the Opening GUI. 5. Checks if the username and password are in the database. If correct, opens the Main GUI. If incorrect, displays error message

2.4) Sequencing

This section describes the flow of events in your program. A flowchart is often the easiest way of representing sequencing in a program; you could also use pseudocode or linked GUI screenshots. Essentially what actions on a screen lead to the opening up of other screen – is the flow logical and useful.? Overleaf are some examples from other projects that you may find useful.

2. SEQUENCING





2.5) Class Design

This section deals with the class design of your NON-INTERFACE CLASSES ie the design of your object classes, array classes and database class (which ever is relevant). Each of the classes you list must have the following information:

- Class name
- Attributes (including types and access modifiers)
- Methods (including parameters, return types and access modifiers)

You must not list your GUI classes in this section. Here only your working code is important. Here you should have a method for the features listed in your System Specification Document. For example if you have a feature called "Add a Client" then you will have a method somewhere in this section called 'addNewClient" which takes the details of the new client as its parameters. This section is important and is worth a significant amount of marks.

Example:

Class - Student

Fields	**Description**
- int ID	Unique number assigned to student
- String firstName	Stores' a student's first name
- String lastName	Stores' a student's last name
- String mother	Stores' a student's mothers name
etc	etc
Methods	**Description**
+ getID( )	Returns student's unique ID
+ String getName( )	Returns student's
+ String getGrade( )	Returns student's grade
+ String getFather( )	Returns student's father's name
+ String getMedical ( )	Returns student's medical issues
+ String toString( )	Returns selected concatenated student fields as per the design of the toString method.
+ public String toStringns( )	Returns student's information in a form that can be easily used anywhere in the program

Array class - Student Manager

Fields	**Description**
- Student[ ] learner = new Student [1000]	Array of objects of all students
- String heading	Stores the heading in the display
- int size	Stores the number of students in the array
- listsize	Stores the number of students in the array that is currently being displayed.
etc	etc
Methods	
+ String [ ] displayFields ( )	Returns an array with all the students and their information.
+ String [ ] getStud(int studid)	Receives an ID, searches for the student with the same ID and returns their information in an array.
+ String [ ] getStudName (String studname)	Receives a name or part of a name and returns all the students that have the same characters as the received name. Returns an array.
+ String [ ] sortGrade(int grade)	Receives a number and returns all the students that are in that grade in an array.
+ void makeRegister( )	Writes all the students names and grades into a text file for printing
+ int getListSize( )	Returns the number of students.

Class Father

Fields	**Description**
- int ID	Unique number assigned to student
- String firstName	Stores' fathers first name
- String lastName	Stores' a fathers last name
- String address1	Stores' part one of the father's address
- String address2	Stores' part two of the father's address
etc	etc
Methods	**Description**
+ getID( )	Returns student's unique ID
+ String getFathersName( )	Returns father's name
+ String getFatherCell( )	Returns father's cell phone number
+ String toString( )	Returns father's information
etc	etc

Array Class – Father manager

Fields	**Description**
- Father[ ] father = new Father [1000]	Array of objects of all Fathers
- String heading	Stores the heading in the display
- int size	Stores the number of fathers in the array
etc	etc
Methods	
+ String [ ] getID (int studid)	Returns the information of a father that matches an ID
+ String displayFields( )	Returns all the information of a father
etc	etc

2.6) Persistent Storage Design

Here you provide a detailed account of your storage design i.e. your database and or text files.

Database - The tables your database is going to consist of . . .
For each table provide

- Field name and types
- Primary key
- Foreign key
- Any relevant formatting of fields in your tables
- Relationship diagram (showing links between tables)
- Sample data for each table

Text files – if you are using text files to store data provide the following

- Names of the text files
- Where they are stored in the folder structure
- Format of the text files including relevant delimiters
- Sample data for each type of text file

Examples:

Database:
Tables:
Student:
Primary key *

Field name	Data Type	Description
StudentID *	AutoNumber	Unique number that identifies a record
FirstName	Short Text	First name of a student
LastName	Short Text	Last name of a student
Grade	Number	The grade of a student
Mother	Short Text	The name of the students mother
Father	Short Text	The name of the students father
House	Short Text	School house of the student
etc	etc	etc

Database:
Tables:
Father:
Foreign Key **

Field name	Data Type	Description
StudentID **	AutoNumber	Student's number
FatherFirstName	Short Text	First name of the student's father
FatherLastName	Short Text	Last name of the student's father
Address1	Number	Address line one of the father
Address2	Short Text	Address line two of the father
CellNumber	Short Text	Father's cell phone number
Email	Short Text	Father's email address
etc	etc	etc

Text Files – some examples and how to record them

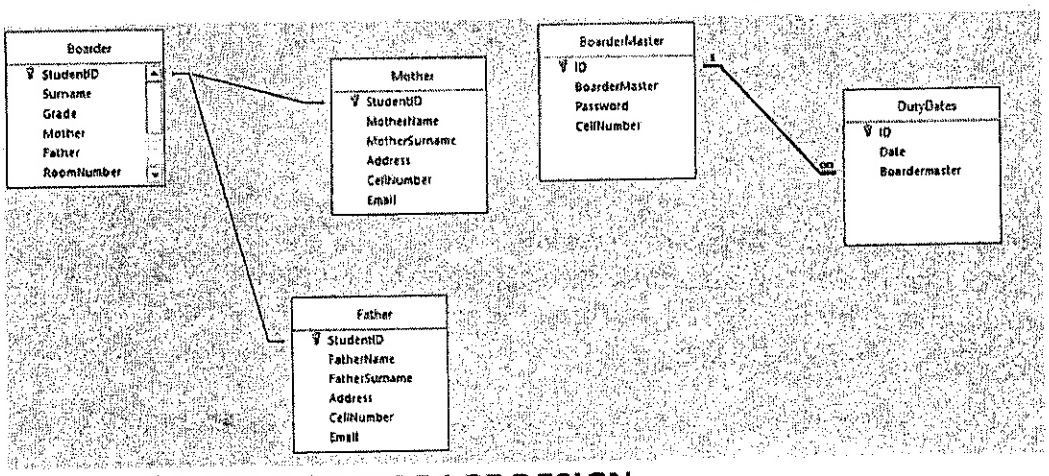
Names:

- Stored in package Pat2018JoeSoap
- Contains students details separted by the # delimiter

Help

- Stored in package Pat2018JoeSoap
- Contains text for the help file

Relationships
Take a screen shot of the relationships in your database
Example:



EXPLANATION OF STORAGE DESIGN

2.7) Explanation of storage design
The reasons behind your storage design. What were the advantages of using a database in your program? If you used a text file, what advantages did it bring?

What type of data do you need to write to the database and what type of data can merely remain in RAM?

Examples: You would write a new student to the database but the results of a name search would merely remain in RAM for temporary viewing.

Document 3: Technical and Testing Document
In this document you have your actual Java code with explanations. Also the results of testing your program when bad data.

1.1 Cover
- Performance Assessment Task
- Technical and Testing Document
- Your full name
- Your grade
- The year.

1.2 Table of contents

1.3) Externally sourced code
You may have used specialty code that you did not write yourself – this is the place to declare it.
Examples
- Connect class. Used to connect to the database. Supplied.

- Special database classses as supplied by your teacher or downloaded off the internet.
- Libraries for sending emails, text messages or working with pdf documents.
- Tools class. Used for displaying formatting. Supplied
- UcanAccess Library. Connects to the database
- Cool code for special effects, audio, transitions

If all code was written by you, you need to declare it eg "I have written all code in this project and have not used externally sourced code"

1.4) Explanation of critical algorithms

Explain two algorithms that are at the heart of your program i.e. usually the one that reads from the database and stores the data in an array of objects. Explain the steps that make up this algorithm e.g. the connect, the loop, the try catch etc.

Explain the code using pseudo code or a flowchart detailing the algorithm (do not copy paste from your Java code for this section)

Explain why you consider this piece of code so important to your project.

1.5) Advanced Techniques

If your program uses code, techniques or methods that are not typical to the IT syllabus explain them in this section i.e. what they do and how they work. Also justified why you included them in your project.

Example: The login screen that requires a username and password for the user.

NOTE: If **you** found the technique to be challenging, even if it was taught in class you can include it here.

1.6) Test Plan and Results

As with any software it needs to be tested with three types of data

1. Normal
2. Extreme
3. Erroneous

Demonstrate and record your testing process with these three types of data. The intention of this part of the program is too see how robust your program is; does it reject garbage, does it present a run time error (exception). Have you used suitable error handling methods to throw and catch exceptions (try . . . catch, using the trim method, suitable if statements making use of String handling methods etc). Does your program present a suitable error message?

NOTE: It is ok if your program does crash. Marks are allocated to the testing processes itself, not the fact that your program did or did not give a run time error.

Example:
The New Student GUI was tested with the following test data

Test	Input	Normal	Extreme	Erroneous
1	First Name	Mary	Longer than 20 char	98nhg*
2	Last Name	Decker	Longer than 35 char	766789
3	Phone Number	0827657876	Longer than 15 char	Jack
4	Grade	12	678865	Grade 12
5	Email	jo@gmail.com	Longer than 50 char	Jo @gmail.com
6	Phone Number	0827658976	Longer than 15	082 765 6547
etc	etc	etc	etc	etc

Record the output from your test results into a table like the one shown below. Take screen shots to show the error dialog boxes as relevant. The number shown below match the tests shown in the table above

Example only:

Test	Normal	Extreme	Erroneous
1	Success	Success	SQL error/Program did not crash
2	Success	SQL Error/ Program crashed	Number format error. Program crashed
3	Success	Accepts input	Program refused input and looped back for the user to try again.