A1

In today's world computers are constantly finding new ways to function as past items; and as observed, many times computers are much more efficient than the said past items. Take, for example, a series of pages: they can only be organized one way at one time, for them to be organized differently you must manually move each page in a different order. This is fine but when it comes to having different information on each page, such as questions that can vary on the same page, one must cut and paste or rewrite and reorder each page. This is an extremely laborious process when you begin to compile large numbers of papers.
The specific problem I am trying to solve is how every year, our UIL team goes to four or five competitions, obtaining a new test every competition, and it become too lengthy to review each test, especially if we're trying to review a certain topic. I need to revise our current method of cutting out questions from the actual test and pasteing them onto a blank sheet of paper, where each question is organized by category and test date. (see Fig. 1)
On top of this, we require additional sheets of paper to display the answers and the order of tests; also, so that this information remains hidden, we couldn't include it in the questions. This mess of papers eventually became unreliable, as they became easily lost, not to mention the space they take up. I decided that, in the theme of computer science, that it would be best if I made a program to aid our studies.

To obtain the needed functions of the program I am attempting to create, I consulted with the head of our UIL team, Mrs. Shoemaker.

Erik Clary: First off, we're going to need a question base and structure; quizzing method and search correct?
Mrs. Shoemaker: Right, we're also going to need a way to display all questions in categories; and options to assemble the tests according to certain guidelines.
EC: Right.
MS: Would we need a question randomization function?
EC: It wouldn't be hard to implement, also have an answer randomizer; just to switch around the orders.
MS: Each question would need a question field, and five answer choices; what else?
EC: The correct answer, and the date or category of the question.
MS: Would the date of the quiz be necessary?
EC: Not really; but which quiz it pertains to would be nice to have if you want to look at the original quiz.
MS: It would also be easier if you could have fields apply themselves without retyping for each question; like from which test the question came from while entering a whole quiz.
EC: I don't follow. You mean entering multiple questions without redoing some fields?
MS: Yes.
EC: That'd become overcomplicated as far as which field you're entering; I say we should limit it to a repeated category or test.
MS: Ok, so the program will have the ability to…
EC: Create quizzes, store questions, order questions by category, test, or difficulty [we need to add that to the list of attributes], display the said questions; and delete questions if needed.
MS: Okay; get to it.

A2

Criteria for Success

After speaking with Mrs. Shoemaker on the needs of the program, I devise that it will need:

A question:

- Needs to have:
  - Question
  - Answers (up to 5)
  - The correct answer : for grading purposes
  - Test Year : to be able to search for
  - Category

The Storage:

- Should be able to display questions in single, by category, test, and create tests by
  - Topic, + number of questions
  - Random, + number of questions
  - Whole test (one way that is just whole, and one that displays questions between a range. Ex: Questions 3-15)
- To be able to find questions
  - By topic, displaying all questions in it
  - By test, displaying all question in the test
- Should be able to store and retrieve questions.

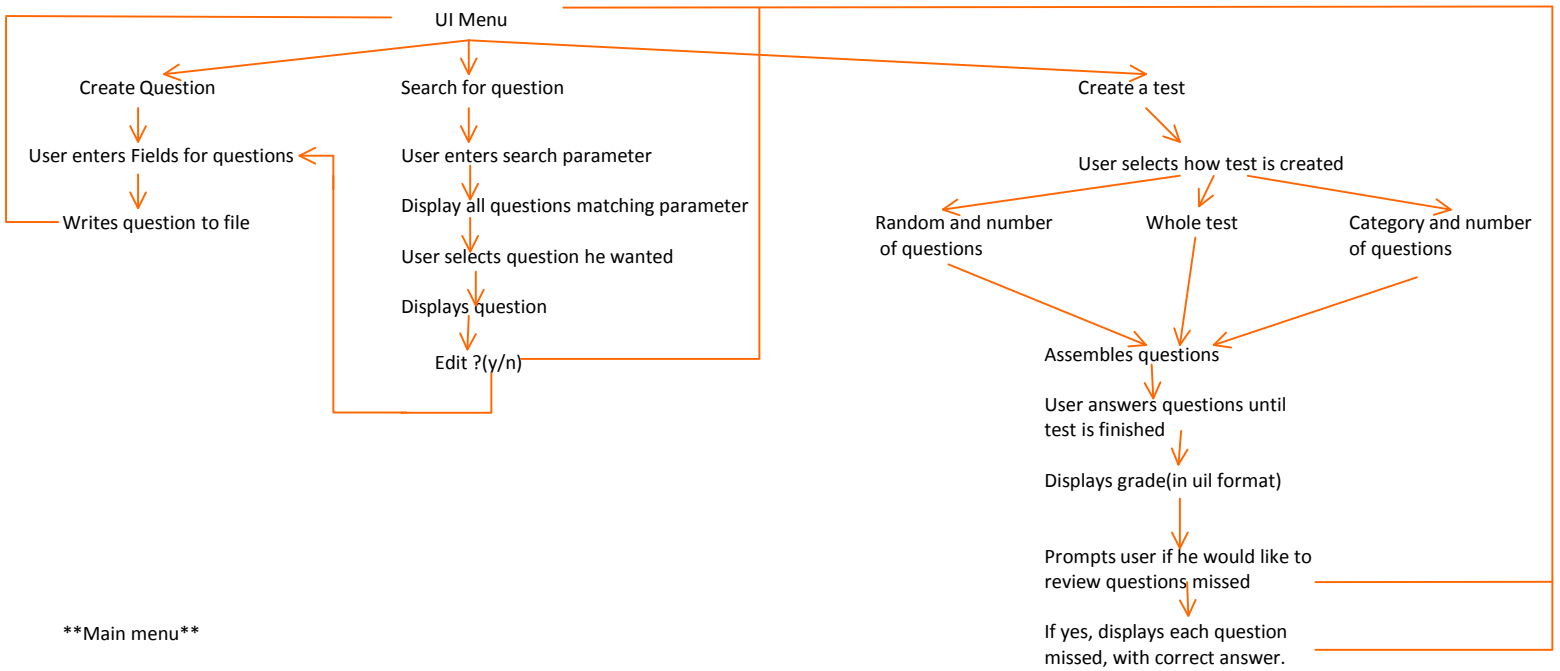
The Display:

- This essential for the end user, without it there would be no input or output.

- Should show score at the end of each test.
  - To be able to judge a user's performance, this way the user will be able to track his or her progress in learning the material.
- Display options for searching
  - This is for ease of the end user, so that she may search for questions to be edited by two different parameters, in case she forgets one.
  - By Test number
  - By Category
- Display and handle input
  - Here the program should be able to display questions, and read the user's input for the answer or choice.
  - Create Question
    - The program should allow the user to create a question which will be stored into a file afterward.
  - Search for Question
    - The program should therefore be able to ask the user if she wants to search for a question by topic or test, then request the appropriate input from the user.
    - Topic
    - Test
  - Create a test
    - The program must be able to compile a list of questions based on what the user requests, having referenced the interview, I infer that the end user wants to be able to take three different branches of tests, by topic, random, and by test year.
    - Topic(##)

        - User created topics
    - Random(##)
    - By test (year)
- Exit
    - Finally, the program must be able to exit itself without causing a crash or having the user to force quit.

A3



Main menu

1. Create Question
2. Search for question
3. Create a test
4. Exit

Please enter selection: 1

Question category?
1. Boolean
2. Math
3. Base change
4. New category

Selection: 4

Please enter new category: Output

Category Created

Please enter test year: 2009

Enter fields:

Question: What is the output of: System.out.print("5");
Answer 1: 2
Answer 2: 3
Answer 3: No output
Answer 4: 5
Answer 5:
Which answer was correct? (please input answer number): 4

Category: Output
What is the output of: System.out.print("5");
A. 2
B. 3
C. No output
D. 5

Correct answer: D
Is this correct? (y/n) y

Question saved

Main menu

1. Create Question

Please enter selection: 2

Please fill in all fields, if you do not know a field leave it blank.

Test year:
Categories:
1. Boolean
2. Math
3. Base Change
4. Output

Category?: 4

System has found 1 question(s) that fit your search criteria:

1. What is the output of: System.out……

Which question would you like to view? --> 1

Category: Output
What is the output of: System.out.print("5");
A. 2
B. 3
C. No output
D. 5

Correct answer: D
Would you like to edit this question(y/n)? Y

If anything is to stay the same, leave the field blank.
Category:
Question:
Answer 1:
Answer 2:
Answer 3: 15
Answer 4:
Answer 5: 8

Main menu

1. Create Question
2. Search for question
3. Create a test
4. Exit

Main menu

1. Create Question
2. Search for question
3. Create a test
4. Exit

Please enter selection: 3

How would you like to create the test?
1. Whole test
2. Category
3. Random

Please enter selection: 1

Please select a test year:
1. 2009
2. 2010

Please enter selection: 1

The test will begin.
Press enter to continue.
What is the output of: System.out.print("5");
A. 2
B. 3
C. 15
D. 5
E. 8

Please select your answer: C

The test has ended, please press enter to continue.

Questions correct: 0
Questions total: 1
Score: 0/1; 0%, -4 out of a possible 7 points.

Press enter to continue to Main Menu

Main menu

1. Create Question
2. Search for question
3. Create a test
4. Exit

Please enter selection: 3

How would you like to create the test?
1. Whole test
2. Category
3. Random

Please enter selection: 2

Which category would you like to review?
1. Output
2. String Methods
3. Base Change

Please enter selection: 3

Answer 2:
Answer 3: 15
Answer 4:
Answer 5: 8
Correct answer:

Category: Output
What is the output of: System.out.print("5");
A. 2
B. 3
C. 15
D. 5
E. 8
Correct answer: D

Is this correct? (y/n) y

Question saved

How many questions would you like to review?

Number: 1

The test will begin.
Press enter to continue.
What is the sum of 1001(base 2) and 110(base 2)?

A. 1011(base 2)
B. 55(base 10)
C. 31(base 5)
D. 1111(base 2)
E. 18(base 10)

Please select your answer: d

The test has ended, please press enter to continue.

Questions correct: 1
Questions total: 1
Score: 1/1; 100%, 7 out of a possible 7 points.

Press enter to continue to Main Menu

Main menu

1. Create Question
2. Search for question
3. Create a test
4. Exit

Please enter selection: 3

How would you like to create the test?

1. Whole test
2. Category
3. Random

Please enter selection: 3

How many questions would you like to review?

Number: 2

The test will begin.
Press enter to continue.
What is the output of: System.out.print("5");

A. 2
B. 3
C. 15
D. 5
E. 8

Please select your answer: D

What is the sum of 1001(base 2) and 110(base 2)?

A. 1011(base 2)
B. 55(base 10)
C. 31(base 5)
D. 1111(base 2)
E. 18(base 10)

Please select your answer: d

The test has ended, please press enter to continue.

Questions correct: 2
Questions total: 2
Score: 2/2; 100%, 14 out of a possible 14 points.

Press enter to continue to Main Menu

Main menu

1. Create Question
2. Search for question

3. Create a test
4. Exit

Please enter selection: 4

The system will now shut down, proceed?(y/n) y

System exited.

B1

The questions:

Questions will have their own class with parameters, therefore allowing the encapsulation of multiple fields into one reference object. Questions will have the question itself, answers, the topic, the correct answer, and the test year that it came out of.

#	Question	Answers	Topic	Correct answer	Test Year
1	What is 1+2?	1. 3 2. 2 3. 0 4. 55	Math	1	1997
2	Do you like ice cream?	1. Yes 2. No	General	1	12
3	QUESTION?	1. QUESTION 2. NOT QUESTION 3. MARK.	Random	3	2020

The Test:

Next I will have a test class, used to create a test by compiling questions. The test will be a linked list of unsorted, probably randomized questions. Its constructor will use the File handler class to search and compile a given number of questions along with whichever parameter is passed. The UI class will likely use the Test class to get each question, cycling through the list.

Test{
File topics;
LinkedList questions;

Constructors etc
}

#	Head	Item	Tail
0	Null	Question 1	Question 2
1	Question 1	Question 2	Question 3
2	Question 2	Question 3	Question 4

The UI class:

The UI class will take care of everything displayed to the screen and any user input, it will also be in charge of storing the amount of questions that the user answers correctly.

Main menu

1. Create Question
2. Search for question
3. Create a test
4. Exit

Please enter selection: 1

Question category?

1. Boolean
2. Math
3. Base change
4. New category

Selection: 4

File Handler:

The file handler class should govern reading from and writing to the files, one being the question storage, and the other to keep track of all the topics, for displaying purposes. It should also be able to return single, and lists of questions; allowing the test class to be more efficient.

As far as bytes are stored, all fields are converted into strings, then either filled or limited to the byte size needed. For example, if a user inputs 13372 into the Test Year field, the program will limit the field to 1337 before writing it to the file, similarly, if the year is input to be 132 it will be written to the file as 132~ using ~ as a 'blank' character to satisfy the record size.

Random Access file:

Record (1560 bytes)

#	Head pointer	Question (1000 bytes)	Answers(500 bytes)	Topic(50 bytes)	Correct Answer (2 byte)	Test Year(8 bytes)	Tail pointer
0	Null	What is 1+2?	1. 3 (100 bytes 2. 2 (100 bytes) 3. 0 (100bytes) 4. 55 (100 bytes) 5. None of the above(100 bytes)	Math	2	1997	Question 2
1	Question1	Do you like ice cream?	1. Yes 2. No	General	1	12	Question 3

Sequential file:

- Topic a
- Topic b
- Glue
- Pets
- Misc.

Linked List:

My Linked List class will be a structure which will mainly store the questions. It should have an add method, delete method, get method (by index), and a size method.

Storing the item/data should be relatively easy as the class would just need 3 attributes, each an object. One of the actual object, one of the object before it, and one after it. Traversing the list should be done recursively, but because Linked List is a type of array, it could be done iteratively.

Instead of creating my own data structure, I could've implemented one of java's: ArrayList, LinkedList, Tree, or Hashset. The way I have my Linked List structured below it's a doubly linked, because it should only be traversed in one direction, it would probably be wiser and more memory efficient, keeping track of the head and tail seem to be easier with a doubly linked list.

#	Head	Item	Tail
0	Null	Item 1	Item 2
1	Item 1	Item 2	Item 3
2	Item 2	Item 3	Null

B2

Question Class

String question
String category
String[]Answers
Int correctAnswer
Int testYear

Constructor Question(string q, string c, string[] a, int cA, int tY)
{
    question = q
    category = c
    Answers = a
    correctAnswer = cA
    testYear = tY
}

setQuestion(string q)
    Sets question to parameter

setCategory()
    Sets category to parameter
setAnswer()
    Sets answer to parameter
setCorrectAnswer()
    Sets correct answer to parameter
setTestYear()
    Sets testYear to parameter

String toString()
    Prints category, question, then answers.
        Answer will be printed via for loop

Test Class
File topics
RamFile questionList
LinkedList questions
Int position(a counter used to store position in the linked list)

Public test(string category, int number)
    While number!=0 && not at the end of file
        Read questions from file:: if equals category
        Stores question in linkedList
        Number --
getQuestion

Returns next question according to position, and adds to position

UI Class

File topics
Final int bytesTestYear
Final int bytesCorrectAnswer
final int bytesQuestion
Final int bytesAnswers
Final int bytesTopic

All methods are static, unless defined otherwise.

DisplayMenu()
Displays menu of choices, requests user for input, then returns their request (int).
getInput()
Returns the user's String inputs.
createQuestion()
Prompts user for question fields: topic, question, answers, and correct answer, then calls question constructor after adjusting each field to its byte allocation. Finally, it writes the question to ram using the RAM class.
editQuestion()
Prompts user for which topic his question is in, then displays all questions within that topic. After, the user is prompted for which question he would like to edit, then allows for him to rewrite any field; if he does not wish to rewrite a field, he may leave the prompt blank, and it will not rewrite the field.
createTest()
Prompts user for which kind of test he would like to take, and how many questions he would like. It then calls test constructor, which creates and initializes the test.
getScore()
Returns user's score.

File Handler class

File topics
RandomAccessFile questions
int questionNumber (the number of total records in RAF questions)

writeToRAF(Question)
Writes question to the Random Access File in order of its topic. (Alphabetical, there is no order to the questions themselves, they are only grouped by topic.)

WriteToFile(String)
Writes a new topic to the sequential file.

Question [] findQuestions(String questionYear, String topic)
Finds questions in RAF questions that match criteria, matching first the topic, then the year; loads all corresponding questions to Question[] then returns it. (Linear Search)

String [] getTopics()
Returns an array of topics found in File topics.

Question [] getWholeTest(String testYear)

Finds all questions that match the testYear parameter, then loads them into Question[], then returns question array. (Linear Search)

Question[] getRandomQuestions(int number)

Calculates numberQuestion;
Random rand = new Random();
For(iterator<number)
Int times = rand.nextInt(0,numberQuestion)
[get record*times question record]
Store in Question[]
Return Question[]

Question[] getTopicQuestion(String topic, int number)

Searches for topic, reads all from the topic to an Arraylist, then gets random questions up to the number given, storing them in Question[]
Returns Question[]

B3

C1

C2

Error Handling

FileHandler Class:

```
if(questionOut.size()<number){
        System.out.println("Sorry, but I was only able to find " + questionOut.size()+ " question{s} that fit
your criteria.");
    }
```

Here the FileHandler notifies the user that it could not find the number of questions he has asked for, this is present on lines 139-141 and 195-197.

Question class

```
for(int i=0; i<ans.length; i++){
        if(ans[i].length()>100)
          ans[i] = ans[i].substring(0,100);  //Here I make sure to check the size of the string before
adding/attempting to add, this was much easier here than
        while(ans[i].length()<100){
          ans[i]+="~";
        }
    }
```

Here I check each answer field when constructing a question to limit them to 100 characters so that they do not cause conflicts in the RandomAccessFile. (lines 38-44)

```
question = questionIn.substring(0,QUESTION); //checks to make sure that the parameter does not
exceed is required length.
    answers = ans;
    topic = topicIn.substring(0,TOPIC);
    testYear = testYearIn.substring(0,TESTYEAR);
    correctAnswer= correctAnswerIn.substring(0,CORRECTANSWER);
```

Here I check the rest of the fields for length, limiting them to each of their respective lengths to allow them to be read correctly from the RandomAccessFile.

UI Class

```
switch(Integer.parseInt(getRequest())){
        case 1: UI.createQuestion(); displayMenu(); break;
        case 2: UI.editQuestion(); displayMenu(); break;
        case 3: UI.createTest().runTest(); displayMenu(); break;
        case 4: System.exit(0); break;
        default: System.out.println("Invalid argument."); displayMenu();

    }
```

In this section I use the default case to check for any kind of user error. I make sure to read a string then parse it to be sure that I catch the line feed after pressing enter.

```
catch(InputMismatchException e){
```

```
      System.out.println("There has been an error in your input, please enter what you a prompted for.
\n**Restarting Edit**\n");
      editQuestion();
   }
```
Here is the end of a try-catch statement in editQuestion(). This catches any error in the user's entry and restarts the method if anything happens. Because the question isn't changed until the user input is complete, there is no error created in the file. (103-164 for the entire try-catch, 161-164 for the excerpt above)

```
try{
        out = new Test((String)list.getItem(select),num);
        }
        catch(ArrayIndexOutOfBoundsException e){
           System.out.println("You have requested a Topic that is not listed. Restarting method...");
           createTest();
        }
        break;
```
This statement is incased in a switch statement, if the user selects an array element that is not listed, he is told so and the method restarts. (188-195)

```
default: System.out.println("Not a valid selection."); createTest();break;
```
Here is the end of the switch started at line 176. This makes it so if any choice is incorrect, the method restarts itself.

C3

Usability

Feature	Demonstrated at…	Code located at...
Allow the user to create a question with fields: Question Topic Year 5 answer choices A correct answer	Figure 1	UI class, lines 59-85
Save a question. (write the question to RandomAccessFile)	Figure 1, Figure 2, Figure 5	UI class, lines 82-83, 155
Find a question by both year and category.	Figure 3,	UI class, lines 92-107, 166-184
Edit a question.	Figure 3, Figure 4	UI class lines 92-160
Create a test based on criteria. (Category, Year, and Random)	Figure 6, Figure 7, Figure 8	UI class lines 166-206
Allow user to take the test.	Figure 6, Figure 7, Figure 8	Test class lines 69-96
Display user's score.	Figure 6, Figure 7, Figure 8	UI class lines 212-217.

D1

```
Output - UILQuestionaire (run)
run:
**Main Menu**

1.        Create Question
2.        Edit Question
3.        Take a Test
4.        Exit
Please enter selection:  1

Enter Question: What is the sum of 1001 (base 2) and 101 (base 2)?

Enter Topic: Binary

Enter Test Year: 2008

Enter Answer 0: 1000 (base 2)

Enter Answer 1: 1110 (base 2)

Enter Answer 2: 23 (base 10)

Enter Answer 3: 1A (base 16)

Enter Answer 4: 16 (base 8)

Enter number of the correct answer: 1
**Main Menu**
```

Figure 1 - Main Menu is printed, user is prompted for input, then the user creates a question. This question (and topic) are written to their respective files,

```
Binary~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~2008~~~~What is the sum of 1001 (base 2) and 101 (base 2)?~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Figure 2 - What is in the RandomAccessFile after Figure 1.

```
**Main Menu**

1.        Create Question
2.        Edit Question
3.        Take a Test
4.        Exit
Please enter selection:  2
Please enter the topic of the question you are looking for: Binary
Please enter the year of the question you are looking for: 2009
Search complete.
0. What is the sum of 1001 (base 2) and 101 (base 2)?
Enter the number of the question you would like to modify:
```

Figure 3 - User now begins to edit a question, this starts with loading all questions that fit either the topic requested or year. Note that even though the question entered in Figure 1 has year 2008, the program still finds the question when it matches the topic.

```
**Main Menu**

1.        Create Question
2.        Edit Question
3.        Take a Test
4.        Exit
Please enter selection:  2
Please enter the topic of the question you are looking for: Binary
Please enter the year of the question you are looking for: 2009
Search complete.
0. What is the sum of 1001 (base 2) and 101 (base 2)?
Enter the number of the question you would like to modify: 0
Please enter the new field when prompted, if you do not want to change the field, simply press the enter key.
What is the sum of 1001 (base 2) and 101 (base 2)?
Enter new question:
Binary
Enter new topic: Numbers
2008
Enter new year: 2009
0. 1000 (base 2)
Enter #0 answer:
1. 1110 (base 2)
Enter #1 answer:
2. 23 (base 10)
Enter #2 answer:
3. 1A (base 16)
Enter #3 answer:
4. 16 (base 8)
Enter #4 answer:
1
Enter new correct answer:
```

Figure 4 - User finishes editing the question. He does not wish to change the answers or the question, only the year and topic.

```
1  Numbers~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~2009~~~~What is the sum of 1001 (base 2) and 101 (base 2)?~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Figure 5 - Contents of RandomAccessFile after Figure 4.

```
run:
**Main Menu**

1.        Create Question
2.        Edit Question
3.        Take a Test
4.        Exit
Please enter selection:  3
Please select the type of test you would like to take:
1. By Category
2. By Year
3. Random
1
Which category would you like to review?
0. Binary
1. Strings
1
How many questions would you like? 4
Sorry, but I was only able to find 2 question{s} that fit your criteria.
Strings
What is printed by the statement: System.out.prinln("Hello World");?
1. Hello World
2. "Hello World"
3. HelloWorld
4. Nothing; there is a runtime error.
5. Nothing; there is a syntax error.

Please enter your anwer: 5
Strings
What is the output of: System.out.println("\n QWERTY \n \\");?
1. (new line) QWERTY
2. QWERTY
3. QWERTY \
4. (new line) QWERTY (new line) \ (new line)
5. There is none due to a syntax error.

Please enter your anwer: 5
Questions correct: 1
Questions total: 2
Score: 1/2; 50.0%
2 out of a possible 12 points.
Press enter to continue to the Main Menu.
```

Figure 6 - The user now takes a test over the category of Strings. The test then gives the user his score, answering correct on the first question, and incorrect on the seconds yields a 50% or a 2 point score on the UIL test.

```
run:
**Main Menu**

1.        Create Question
2.        Edit Question
3.        Take a Test
4.        Exit
Please enter selection:  3
Please select the type of test you would like to take:
1. By Category
2. By Year
3. Random
2
Which year would you like to review? 2009
How many questions would you like? 3
Sorry, but I was only able to find 1 question{s} that fit your criteria.
Numbers
What is the sum of 1001 (base 2) and 101 (base 2)?
1. 1000 (base 2)
2. 1110 (base 2)
3. 23 (base 10)
4. 1A (base 16)
5. 16 (base 8)

Please enter your anwer: 2
Questions correct: 1
Questions total: 1
Score: 1/1; 100.0%
6 out of a possible 6 points.
Press enter to continue to the Main Menu.
```

Figure 7 - The user now takes a test by year, though he requests for four questions, he has only entered one of year 2009, so the program explains this and only prints that question.

```
**Main Menu**

1.        Create Question
2.        Edit Question
3.        Take a Test
4.        Exit
Please enter selection:  3
Please select the type of test you would like to take:
1. By Category
2. By Year
3. Random
3
How many random questions would you like to review? 5
Strings
What is the output of: System.out.println("\n QWERTY \n \\");?
1. (new line) QWERTY
2. QWERTY
3. QWERTY \
4. (new line) QWERTY (new line) \ (new line)
5. There is none due to a syntax error.

Please enter your anwer: 4
Numbers
What is the sum of 1001 (base 2) and 101 (base 2)?
1. 1000 (base 2)
2. 1110 (base 2)
3. 23 (base 10)
4. 1A (base 16)
5. 16 (base 8)

Please enter your anwer: 2
```

```
Strings
What is printed by the statement: System.out.prinln("Hello World");?
1. Hello World
2. "Hello World"
3. HelloWorld
4. Nothing; there is a runtime error.
5. Nothing; there is a syntax error.

Please enter your anwer: 5
Strings
What is the output of: System.out.println("\n QWERTY \n \\");?
1. (new line) QWERTY
2. QWERTY
3. QWERTY \
4. (new line) QWERTY (new line) \ (new line)
5. There is none due to a syntax error.

Please enter your anwer: 4
Numbers
What is the sum of 1001 (base 2) and 101 (base 2)?
1. 1000 (base 2)
2. 1110 (base 2)
3. 23 (base 10)
4. 1A (base 16)
5. 16 (base 8)

Please enter your anwer: 2
Questions correct: 5
Questions total: 5
Score: 5/5; 100.0%
30 out of a possible 30 points.
Press enter to continue to the Main Menu.
```

Figure 8 - The user takes a randomized test, but requests more questions than he has entered, the program handles this by returning to the beginning of the RandomAccessFile and searching for the random questions again.

D2

My program did fulfill all needs discussed in A2; however, there are a few points that should've been added to the program.

I never thought about deletion, the nature of this program is to be a database that would recall what it is given, and I allowed for modification of the questions if the user where to enter something wrong. I should have included a way to delete from the Random Access File in case a question becomes outdated. In fact, I should have implemented a way to delete entire topics in case they become outdated.

My Random Access File and Sequential File should have been ordered by topic and alphabetically, respectively, allowing the user to quickly find a certain topic in case there become too many, and the program to be more efficient in its search for questions. There is also a hardware limitation in that one could create a StackOverflowException by requesting too many questions to be assembled in RAM. Also, when requesting for more questions than have been entered in the random type of test, the questions seem to repeat in the same order, though I'm sure it has something to do with the random class' assembly of random numbers, it would have been better if I re-hashed and randomized the list after its assembly again, which may slow down the program a little, it would help the randomness of the test.

For the end user, I believe that the program is functional and very much able to handle everything we had discussed. The only limitations that I can see are using the '~' character in any question entry and some user errors beyond my control. The program instructs the user how data should be entered, and does handle the data if it is too large or small. Also, there is no recovery system in place, therefore if any data were to become corrupt, the user would have to manually search through the data and adjust it accordingly. In addition, it would've been wise to have a backup set of data files in case something where to go wrong during execution. The user would be given the choice of backing up her data when exiting the program each time, that way if something did go wrong, she could choose not to backup, whereas an automated backup would do it every time the program is exited. If I were to make a smart form, it would require a flag to recognize that there is an error, but sense the program may still read a user's faulty data fine, and it would send no flag. Because this is somewhat of a primitive program by today's standard, the data is easily follow-able to the experienced eye, and because this was designed for a computer science team, further additions should be made by end users and if there are bugs found it could be a learning experience.

In all due seriousness, for those who wish to use this program for other purposes, such as review, quizzes, or other activity, it would've been better to incorporate a GUI based system, this would make the program less of an eyesore and more user-friendly. The program itself has only one flaw still present in the coding, the file location. Due to the Netbeans IDE, it could not find the file unless it was explicitly stated. Therefore, when trying to 'install' this program one would have to manually change the file location to suite their needs. A simple fix could be bundled with the instructions if needed, though the end user I made this for can easily fix this on her own. I do recognize that this is not something the end user should have to do, Netbeans has a certain quirk to finding files within the same folder without the entire path being specified. I tested this against Jcreator where Jcreator does not need the entire path. This may have something to do with the "project" approach of Netbeans but I'm not entirely sure.

Mastery Factors

Adding data to an instance of the RandomAccessFile class by direct manipulation of the file pointer using the seek method	FileHandler at lines 158-161; UI at line 82.
Searching for specified data in a file.	FileHandler at lines 170-197, 113-143; UI at lines 92-101.
Recursion	Node at lines 93-98, 128-141, 162-178, 196-207, 227-233, 240-273 (272), 299-307; UI at lines 27-42.
Polymorphism	Node at lines 128 and 148, 185 and 196; LinkList at lines 32 and 42.
Encapsulation	Question at lines 11-15 (private attributes) and 87-168 (get and set methods).
Parsing a text file or other data stream	FileHandler at lines 82-103, 113-143, 170-197, and 205-228.
Arrays *	FileHandler at lines 42, 87-96, 121-134, and 181-190.
User-defined objects *	Question Class; FileHandler at lines 98, 134, 190, 224; UI at lines 79, 113, and 153.
Simple Selection *	UI at lines 103-150.
Complex Selection *	UI at lines 33 and 171.
Loops *	UI at lines 71, 104, 137, and 175.
ADT Aspect 1	LinkList at lines 16-18, 24-26, 32-35, 42-45, 60-64, 70-72, and 91-93.
ADT Aspect 2	LinkList at lines 32-35, 42-45, and 51-53.
ADT Aspect 3	LinkList at line 62; Node at lines 76, 79, 83.

* indicates an SL mastery factor.