**A QUESTION-RANDOMIZING PROGRAM**

**Introduction**

This program was made to make creating random questions much easier and still identifies the correct answers and is able to display results to the user. This program has been coded to easily add any number of questions easily and making few changes to the source file such as the course title and code, number of questions, index number of registered students.

**Method**

This program was made with functions. These were either user-defined or in-built. We have functions like inarray(), inmat(), scored() and xrandi() which are user-defined. And we also used in-built functions like strcmp(), randi(), disp(), fprintf(), input(), fopen(), fclose() and lower().They all played specific roles in the making of the program. Functions make lines of programs smaller as there is no need recreating codes for similar tasks to be performed. Selection statements, loops and logics also came in handy. We will discuss the how these functions helped create this program.

First, we created functions we might need in the course. The xrandi() function was created to takes care of random number generation making sure the numbers don’t repeat themselves unlike randi(), till the number of random numbers we need is generated. It calls another function, the ‘rng’ function with ‘shuffle’ as the argument, so that at every given time the numbers generated will differ with time even when matlab is restarted. It receives one input argument, that’s the number of random numbers you want, from one to that number. It returns an array containing the numbers. The inmat() function is used to check is a number occurs in a matrix, this is useful when we want to check if an index number can be found inside a matrix of numbers. It receives two input arguments. First is the number and second is the matrix you will be dealing with. It returns a 1 (that is true if the number is in the matrix) or 0 (that is false if the number can’t be found).The scored() function receives two arrays as input arguments, compares them and returns the number of corresponding array elements that are the same. This particular function is helpful in determining the score of the candidate by comparing two arrays one containing the answers and the other is an array of the user’s answers chosen.

The in-built as we said earlier helped a lot, taking away lots of stress. Some work very similar to the user defined functions. The strcmp() function compares two string arrays and returns 1, indicating true and 0 indicating false. We used it a number of times to check the validity of input data especially. The randi() function receives positive scalar integer values (that is the maximum value) and generates random numbers with it from1. The disp() function receives string values to display. The fprintf() function formats data and displays the result on the screen. It is used also to write data into a text file. The input() function is used to receive input from the user. For example, we used this function to get the candidate’s index number and the course code he will be using. The fopen() function is used to open a text for use. It takes either one or two arguments. The first one specifies the name of the text file and the second the mode (either w, r, a, w+, r+, a+, A, or W with each having their own use). The fclose() function is used to close a text file after it has been opened. It receives a file ID as the input argument or “all” which closes all opened files. We used the lower() function to make the candidates answers lowercase to correct situations where by the users inputs both uppercase and lowercase letters.

So with those all those functions and many more made ready, we moved on to create our script files. We created four scripts for this project. These scripts have their own roles that they play and are brought together to perform their task one after the other. We have expounded their uses below.

First let’s start with the check\_validity script file. This m-file checks the user’s input data and determines how valid they are before the questions can be displayed. And this is how it works; it contains the course code and the index numbers. It extracts the index numbers from a text file named index\_nums1.txt and index\_nums2.txt, where the former contains registered index numbers for the English course and the latter contains registered index numbers for the Biology course. It receives inputs from the user and saves them to the variables getcode (has the input course code) and getindex (has input index numbers). And for a particular course code and index number displays some strings to confirm the type of course examination they are writing. If index number and course code entered has a problem the program tells the user to check and enter the data correctly.

The other script named query is where the questions are displaced to the candidate in a random manner. This is how the program also works; so after the user has entered the necessary data correctly the program is allowed to start running (note that the program won’t run till data enetered in check\_validity script is correct). For a particular course code entered the program displays a particular set of questions either for Biology or English.

The my\_score m-file displays to the user his score in percentage if he really wants to see them. The script contains the original answers for each set of questions. After the execution of query ends the user is asked if he would like to view his score with a ‘y’ or ‘n’ reply with y meaning and n meaning no. The script makes explicit use of the user-defined function, scored which returns the number of similar corresponding elements in two arrays, one the original answers and the other the answers received from the user. A ‘n’ reply ends the program with a string.

There are two extra script files we would like to describe. One is the goto m-file and the other is continuefrom m-file. The former, when run will enable the student move to a particular question he would like to answer again. And the latter helps the candidate start from a question he would like to answer again to a particular question.

**Conclusion**

Finally, the program randomizes questions for the user taking the exams. It’s a recommended program for uniquely randomizing sets of questions based on the number of students who are to write the exams. It also prevents students who index numbers are not in the database from partaking in the exams, thereby ensuring that there are no examination mal-practices.

NAMES OF MEMBERS INDEX

1. OFORI EMMANUEL 9344917
2. OPPONG-NKRUMAH DAVID 9345817
3. OPPONG FELIX KWADWO 9345717
4. OFORI GERALD BOSU 9345117
5. OPOKU DENNIS 9345617
6. OFORI EVANS GYAN 9345017
7. OKOLO OSHOKENOYA EMMANUEL 9345317
8. OKRAH JEFFREY KWAKU 9345417
9. ONWUDINJO KELVIN CHUKWUDI 9345517
10. OSEI-MANU EPAPHRAS DWAMENA 9345917
11. OSMAN SULEMANA MALTITI 9346017
12. OTENG DUAH NANA WIAFE 9346117