George Mason University

SWE 432 Design and Implementation of Software for the Web J. Offutt

**Final Exam, 19 May, 2020**

Your final exam will be a tech-challenge programming project.

In the next 2.5 hours, you will build a tiny web application. I am providing specifications of minimal required elements (***MRE***), and specifications of several additional optional required elements (***ORE***).

Each element will be graded on completion and on quality of the work. Completion (it works) will count for approximately 2/3 of each element, and quality will count for the other 1/3. The quality evaluation will include usability, design, and quality of the code.

The project will be worth 30 points to match the 30% of the final grade allocated to the final in the syllabus.

**RESOURCES**

Each student must work individually—no collaboration, no discussion during the exam period, and no help with design or debugging. I will be available for questions. You may use any course materials, including books, the course website, slides, examples, piazza, etc. You may also use the internet, including any libraries, packages, code, or algorithms, as long as they are free or open source. You may **NOT** submit any complete solution you find on the internet or any solution that includes formatting or the UI. That is, you must completely design the UI yourself.

**TECHNOLOGY**

You are required to build at least one front-end component that runs on a client in a browser, and at least one back-end component that runs on a server. The back-end software component should process inputs from the front-end component. Beyond that, you may use any technologies we discussed in the course. Note that the use of certain technologies are explicitly part of the *ORE*s.

**SUBMISSION**

1. You may deploy your running software on github-heroku, AWS, or anywhere else. The important part is that I must be able to run your program through the URL you provide.
2. Submit your final exam by putting all files in a folder named “*MasonID*-swe432,” and zip the contents into one zip file. You should replace “*MasonID*” with your Mason ID, that is, your email address. Thus, my submission would be named “offutt-swe432.zip”. Include the following files:
   1. Page three of this document, with all information filled out.
   2. All source files, including .java, .js, .css, and anything else that you use. (*Exception*: If you use React, do NOT INCLUDE the *node\_modules*/ folder. It takes an unreasonable amount of space.)
3. Submit your zip file into my dropbox file request. You will be able to place your file, but not see it or change it after you submit. If you need to submit a correction, add “-v2” to the file name.

Submit your file through this URL: **XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX**

**ACCOMMODATION**

If you need accommodation, either as specified by the DRC at the beginning of the semester, or due to the pandemic and online nature of the final, contact me by email individually. Online accommodations could include internet access problems or illness. If your internet crashes during the exam period, contact me as SOON AS POSSIBLE through email.

**SPECIFICATIONS**

**Minimal required elements (MRE)**

1. (**9 pts**.) Your web app will accept a string that represents a boolean predicate that has boolean variables and logical operators, similar to ‘if’ statements in programs. Examples include: “**A OR B**”; “**x && y**”; “**M and N or Q**”; and “**today | tomorrow**”. Note that you are expected to design the syntax and format, including the symbols used for the logical operators. For MRE credit, your web app only needs to handle ANDs and ORs, not parentheses, relational expressions (“x>0”), or other logical operators.
2. (**9 pts**.) Your web app will process the string to create and display to the user a complete **truth table** for the predicate. “*AND*s” and “*OR*s,” using syntax you choose, are *logical operators*. Clauses in the predicate evaluate to boolean values, and are connected by logical operators, but do NOT include logical operators. Thus, the above 4 examples have 9 clauses in total: **A**, **B**, **x**, **y**, **M**, **N, Q**, **today**, **tomorrow**. The truth table for the first predicate, “**A OR B**,” has 4 rows: *true-true*, *true-false*, *false-true*, and *false-false.*

Note that you are to design the input and screens, design how the truth tables are displayed, and design and write code to find clauses within the predicates. I am intentionally **not** giving you requirements about how the UI will look or behave.

**Optional required elements (OREs)**

If you submit a program with MREs only, your maximum grade on the final exam will be 60%, which will count as a C grade (passing) for the exam. Implementing OREs can increase your score to as much as 100% (30 points), but no higher.

Choose **no more than four** of the following OREs:

1. (3 pts.) Input validation—report invalid strings.
2. (3 pts.) Allow parentheses in the predicates, for example, “**((A or B) and C)**”.
3. (3 pts.) Evaluate the predicate for each combination of truth values. For example, for the predicate “**A OR B**,” the result values would be *true*, *true*, *true*, *false*; using the same order as the rows in MRE **#II** above.
4. (3 pts.) Add an additional screen, after the truth table is displayed, to evaluate specific truth-values. The user can assign truth-values to the clauses, and your app will return the value of the predicate. Your software will need to maintain state through the multiple requests (any method is allowed).
5. (3 pts.) Include an additional logical operator—*exclusive or.*
6. (3 pts.) Allow multiple syntaxes in the input box for logical operators (for example, “*&*”, “&&”, “*AND*”, “*and*”).
7. (3 pts.) Allow constant binary values to be included in the predicate (for example **“((A and (TRUE or C))**”).
8. (3 pts.) Give the user the option to display the truth-values in the truth table in different formats, such as “*t-f*,” “*T-F*,” “*1-*0,” “*true-false*,” etc.

The maximum score on the final tech challenge exam is 30 points. I am not allowing extra credit on the final, so I will only consider **four** OREs. In your submission, you must tell me which OREs you attempted—up to four. If you attempt five, and earn 2 points on each, you will earn 8 points total, not 10.

The following algorithm may be helpful, although it is not required that you use it.

|  |
| --- |
| \* A recursive algorithm to print a truth table of 1s and 0s.  \* ***N*** is the number of clauses, or columns, in the truth table.  \* ***index*** should be zero on the first call  \* ***truthVals*** starts as an empty array of integers of size ***N***  printTruthTable(integer N, integer index, integer array truthVals) {  if (index == N) {  for (i=0; i<N; i++)  print(truthVals[i] + " ");  print(newline);  } else {  for (i=0; i<2; i++) {  truthVals[index] = i;  printTruthTable(N, index + 1, truthVals);  }  }  } |

**Fill out and submit this page with your final exam**

Your name as on the roster: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Your GMU email address: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The URL where I can run your program: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(IMPORTANT! Test the access to your program through the URL before submitting.)

List the technologies that you used in your final exam program:

Which of the following OREs are you submitting for grading (do not check more than 4):

[ ] 1. Input validation

[ ] 2. Parentheses

[ ] 3. Evaluating the predicate

[ ] 4. A screen and code to evaluate specific truth-values

[ ] 5. Exclusive or

[ ] 6. Multiple syntaxes in the input box for logical operators

[ ] 7. Constant binary values in the predicate

[ ] 8. Different formats to display the truth-values in the truth table

“*I swear upon George Mason University’s Honor Code that I have not discussed this exam or shared any part of my program with anyone but the instructor, and will not discuss this exam or share any part of my program with anyone who has not yet completed the exam*”:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_