Project Specs, Control Flow Graph and Debugging

This activity is an exercise for the topics reviewed in Weeks 2 and 3, including Specs, Control Flow Graph and Cyclomatic complexity and Debugging of the code using debugger.

After completion of this group activity, you will be able to:

* produce Control Flow Graph of a minimalistic application.
* calculate the Cyclomatic complexity of the code provided.
* run the code in GitHub Codespaces and debug in the debugger of the browser.
* compare the Specs of the software with the Production.

## Process of the Activity

In this activity initially you will receive the Specs and the Production code of a simple minimalistic web application: <https://github.com/umassdgithub/Activity-2-CIS-470-SP25>

You will need to consider both the “Product Code” and the “Requirements” in the debugging of the software.

A diagram of a program behavior

AI-generated content may be incorrect.

You will receive some random test cases (in Readme.md file inside the repo) that your code should be tested against it (note that the test set is not complete. More about it in W4.)

The submission should be a compressed folder with each deliverable identified clearly in a separate folder.

### Task 1: Make your own repository of the project (I have enabled Fork as well). (20 points)

* Fork the repository in the link
* Option 1: Use Codespaces in GitHub to run the code.
* Option 2: Use local editor to work on the code:
  + Clone your own copy of repo to your local disk. git clone uri
  + Review the content of the cloned Repo in your local folder.
  + Note that the cloned data folder contains following files and folders:
    - .devcontainer
    - .github/workflows
    - favicon.ico
    - index.html
    - readme.md
  + To make changes appear in your online repo
    - Use: git add . to stage the code
    - Use: git commit -m “message” to commit the code
    - Use git push -u origin main to push the code to your GitHub repo.

**Deliverable 1:** Screenshots (2 or more) of Codespaces (port-forwarding should be active in the screenshot)/VS Code, terminal and Live Preview.

The screenshots should include following details:

1. “Ports” showing the port forwarding is running (in case of option 1)
2. Files and folders showing the files names in the project in vscode
3. Name the screenshots properly (e.g. SC\_Del1\_01.png)

A computer screen shot of a program code

Description automatically generatedA screenshot of a computer

Description automatically generated

### Task 2: Identifying Logical Errors Using Call Stack and Variable Values (20 points)

Please use the Test Cases provided in “Readme.md” to identify the errors. Make sure to use Breakpoints, Call stack and watches to identify the errors in the code. By adding the **debugger** function to the code, you can debug the code (refer to the slides).

See this link: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/debugger>

**Deliverable 2:** When you figured out the error using variable watch, and call stack take a screenshot from the debugger.

The deliverable 2 compressed folder should include:

1. List the errors and their line number, mention the error and its effect in one line. (this can be shown in screenshots e.g. SC\_Del2\_01.png) or just textually listed. (10/20)
2. Describe how breakpoints and call stack have helped you find the error in in a few sentences. (10/20)
3. Screenshot should show the call stack, local and watched variable. You can include multiple files to cover all.

### Task 3: Fixing Calculation Errors (20 points)

Now that you have figured out the errors, it is time to fix the error, accordingly, run all the test cases “manually”, and make sure the results match with the test results provided.

\* Note that the automated testing will be introduced later during the semester (during unit testing lectures).

**Deliverable 3:**

1. List of fixes applied to the code in a text file (  
   example:  
   1) in function foo: if(i<2) 🡺 if(i<=2)  
   2) in function foo: + operator changed to - operagor  
   …  
   )
2. Screenshot of GitHub repo
3. Upload the compressed project folder in myCourses.

### Task 4: Control Flow Graph of the Code (30 points)

**Deliverable 4:**

* 1. Control Flow Graph of the function (only this function):

**function federalTax(taxableIncome,filingStatus="individual")**you should start with Block Diagram First: include the digital version of your control flow graph in the final deliverables folder (you can take picture from the hand drawn diagram). The graph can be drawn either on paper or any other drawing software (e.g. [miro.com](https://miro.com/))

**(note that this will be a complex block diagram)**

* 1. Use the Block Diagram/CFG to calculate the Cyclomatic complexity

### Task 5: Describe your method in finding errors (10 points)

**Deliverable 5:**

1. In a few sentences describe the process you used to find the errors in the code.
2. How different the specifications of the code are compared to the implementation.
3. Suggest the changes that should be made to improve the clarity of the Specifications of the code.

The submission should be a compressed folder with each deliverable identified clearly in a separate folder.