Programming Fundamentals Final

Rather than a final exam for this class, you'll be asked to submit a final project.

You can choose to come up with your own idea for the project, or you can ask me for an idea you can use if you're really stumped. If you do your own idea, you must make sure your submission meets all of these specifications and requirements:

1. Your program must take in at least one input from the user
2. Your program must use some sort of complex decision construct such as a nested IF statement or a Switch statement.
3. Your program must use some sort of loop that you write. In week 7 we'll cover loops quite a bit.
4. Your program must use functions to organize its processing steps and separate processing from input and output.
5. Your program must display some sort of output that uses the input.

# Final Project Ideas:

Text adventure games, business software, decision-making programs, and even graphical games and interactive experiences are all acceptable projects.

However, the following programs are not acceptable as their solutions are easily available on the Internet or they are too simplistic for this class:

1. Guess The Number
2. Rock / Paper / Scissors
3. Maze generators
4. Any program you've been assigned in this course already or any program that is simply a rewrite or modification of a previous assignment.

Any questions on any of these? Check with me before going forward.

Your project should be unique, challenging, and should demonstrate your mastery of the competencies of this course. It should be something you can be proud to show off during an interview. But it should also be something you can accomplish at your skill level.

You'll submit a description of your project and get it approved before you can start your project. That way we can be sure it meets the expectations and that you can make it happen.

If you're not sure if your project fits the scope of this final, come talk to me.

It's in your best interests to choose your own final project. The program I will provide is very challenging and may be more difficult than something you may devise on your own. It also won't really give you the opportunity to build something you really like.

# Project Specifications

The final is worth 300 points and will test your knowledge of the course competencies. It will also demonstrate to me that you are ready to move on to the next class.

Of course, you'll need to do more than just provide code. Here are the complete specifications for the project:

1. You must provide a problem statement - a summary just like the ones you've seen in all the assignments we've done in class every week. This should describe your program's purpose and what it will do. (10pts)
2. You must identify the nouns and verbs in the problem statement just like you've done before. (10pts)
3. You must create a defining diagram filled out appropriately with the inputs, processing steps, and outputs. (10pts)
4. You must create a flowchart for your program using the symbols we used in class. The flowchart must accurately represent the flow of your program. (40pts)
5. You must create pseudocode of your program. The pseudocode must contain the actual algorithms and steps your program will contain. Be specific. (40pts)
6. You must create at least 4 test plans with expected outputs and actual outputs. (40 points)

These items will be worth 150 points.

The other 150 points will be the successful implementation of your program in JavaScript. In order to receive any of these 150 points, your program must execute without any syntax errors. That's right - a program that doesn't run will earn a zero score for this part of the final.

This sounds strict, but the purpose of this class is to demonstrate that you can write code that works. You've learned how to debug your code, so be sure to put those skills to use. In addition to providing working code, you'll be graded on the following:

1. You'll ensure all code follows the class coding standards. You'll use meaningful variable names and apply comments where necessary, ensuring that you have comments at the top of your program explaining what it does and who wrote it. This will be 40% of the grade for this portion of the code.
2. You'll make sure all of your code is indented. This includes pseudocode and JavaScript code.
3. You'll separate your JavaScript from your HTML file. You'll learn how to do this in week 7.
4. You'll make use of decision processing statements, loops, functions, and other programming constructs effectively and efficiently.
5. You will properly handle type casting and precision issues.
6. You'll ensure that you reduce as much duplication as possible, using repetition or other concepts to avoid writing the same code multiple times.

That's it. Now, here's how you should structure your files for submission.

## Grading Criteria for program

Provided that your program runs, I will then apply the following grading rubric to your code:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Does not meet  expectations.  Work is missing. | Partiallly  meets some expectations: work is  incomplete. | Work  minimally meets  expectations. Grasp of the learning is  thin, critical thinking iis  not  demonstrated well, and  there are many areas of detail  lacking. | Work meets  criterion  adequately; and shows grasp of  the  learning, critical  thinking skills, or attention to detaill. | Work  exceeds criterion,  incorporates concepts  not  assessed  previously,  shows depth in insight,  grasp of the learning,  critical thinking  skills, and  attention to detail. |
| Program Logiic You  produce code that is logically and syntactically correct in accordance with a specification or requirements. | 0 | 10 | 30 | 40 | 50 |
| Program Efficiency  You produce code without unused variables, functions, etc. You produce code that follows the DRY Principle. You use functions, | 0 | 10 | 30 | 40 | 50 |

variables, and loops to break down code into reusable components.

Program

Readability You produce code that is readable and self- describing, with clear variable names, and proper indentation. You use functions to encapsulate complexity to improve readability. Spelling errors and grammar errors are to a minimum.

0 10 30 40 50

## Grading Criteria for documentation materials

Does not meet

expectations.

Work is missing.

Partiallly

meets some expectations: work is

incomplete.

Work

minimally meets

expectations. Grasp of the learning is

thin, critical thinking iis

not

demonstrated well, and

there are many areas of detail

lacking.

Work meets

criterion

adequately; and shows grasp of

the

learning, critical

thinking skills, or attention to detaill.

Work

exceeds criterion,

incorporates concepts

not

assessed

previously,

shows depth in insight,

grasp of the learning,

critical thinking

skills, and

attention to detail.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| Problem statement is detailed and expresses the problem being solved | 0 | 5 | 8 | 9 | 10 |
| Nouns and verbs in problem statement are accurately identified | 0 | 5 | 8 | 9 | 10 |
| Defining Diagram lists the inputs, outputs, and processes accurately and in the correct sequence | 0 | 4 | 8 | 9 | 10 |
| Pseudocode adequately describes the problem in specific detail. | 0 | 20 | 30 | 35 | 40 |
| Flowchart uses appropriate symbols and accurately shows the process of the program in specific detail | 0 | 20 | 30 | 35 | 40 |
|  |  |  |  |  |  |

Test Plans are complete, extensively test your program.

0 20 30 35 40

# Exceeding expectations?

After reviewing the rubric, you're probably wondering what it means to exceed the expectations. Well, that part is really up to you. But here's an idea.

Joe writes a simple calculator program and does 4 test plans that use different input values. He writes just enough pseudocode to explain the problem. He then writes his program, using only one IF statement and one loop. He wraps his entire program in a function so he meets the "must use a function" requirement.

Joe's going to get a C on the final. He'll pass but it's just meeting the expectations.

Mary develops a program that pairs wine with food. She develops 8 different test plans, including plans that deal with bad or missing user input. She creates detailed pseudocode that includes the logic needed for string concatenation.

When she writes her code, she organizes components into functions. Her code protects against user input errors, asking the visitor to retry if they enter something wrong, without exiting the program. Her program has minimal duplication, no validation issues, or logic errors. Her user interface consists of prompts and document.write statements.

Mary's going to get a A- for her program. It's very good work but it could be better.

Kevin follows a similar approach to Mary, writing numerous test plans, several flowcharts, and great pseudocode. But he decides that he wants to use HTML forms as the interface for his application. He uses jQuery to code up the interface. In addition to his well-formed HTML that validates, he's also used CSS to style the page.

Like Mary, Kevin went beyond the requirements of final project. He's getting an A.

Each project is individually assessed. You are not competing with your classmates.

However, note that your basic program's functionality has to work. You won't get the exceeds expectations score if you have missing elements, broken logic, or poorly performing code.

# Working on and Submitting Your Project

1. Put all your code in its own folder called yourname\_final where yourname is your CVTC username, to avoid any ambiguity.
2. Under this folder, create a folder called docs .
3. Create one file called design.txt in the docs folder that contains the Problem statement, nouns and verbs, defining diagram, pseudocode, and your test plans.
4. Place a PNG file of your flowchart in the docs folder.
5. Create a file called index.html that contains your HTML. This file should not contain any

JavaScript code.

1. Create a folder called js that contains your JavaScript files. Place all of your JavaScript code within files in this folder.
2. Link your JavaScript files to your index.html file.
3. Test, test, and test your code to ensure it's free from errors.
4. Zip the folder up as yourname\_final.zip and submit it to the dropbox.

# Academic Honesty

Work alone. Do not ask other students for help with this project, and do not show your code to any students. You may ask me for as much help as you need, but do not ask another student.

Do not share code with other students.

Do not use solutions you find on the Internet. It's fine to use sites like StackOverflow to figure out how to solve a particular problem but if you do that, leave a comment with a link to the original code.

However, turning in someone else's completed program as your own is dishonest and will earn you a zero on this assignment. You'll also be brought to the attention of the department chair and the dean. There are many tutorial sites on line. Don't be tempted to do one of those projects.

I reserve the right to ask you to demonstrate your understanding of any code in your project. If you are unable to explain what you did, you may receive a failing grade on the implementation portion of the final.

# Above And Beyond

Many students have implemented object-oriented programming techniques, event handlers, DOM manipulation, and much more into their final projects. While you can do the bare minimum, it's probably not something you'll want to show a potential employer. Several previous semester students have gotten internships based off of the final project in this course, because they went above and beyond what I ask for here.

However, if you do choose to go above and beyond, ensure that everything you're being graded on works first! It would be terrible if you worked hard to create something amazing and ended up turning in a program that failed to work.