Computer Programming Contest

Programming Competition

Contestant Packet

April-2025

Programming Guidelines

- All programs must include internal documentation so that they can be supported by others after they are deployed. All classes and methods should fully document the inputs and outputs.
- All programs input should be checked for security and should provide adequate errors in the event of invalid input.
- While solutions can include a compiled EXE, the executable should be re-creatable from the committed code.
- The programs should be self-contained such that everything required to compile and run the program is included in the repository, as well as any documentation required to perform the compilation.
- You may use a database if you choose, but it is not required. If a database is used it can only be used for simple data storage and not itself provide the solution. Please include the scripts to create any database schema as required. Examples of acceptable databases would be Java DB Embedded or MySql.
- You may use standard Toolkits, API's, Frameworks. So long as they are used in supporting the creation of the application and are not the application themselves. Things included in the Standard Packages are allowed, as Java SE includes JDBC, JFC, Junit, JavaDoc. Other packages like Thymeleaf, Typescript may be used as well.
- If you are including external libraries, be sure to package the solution with an Ant, Maven or similar build system to simplify deployment.
- You shall create a separate directory in your Git repo for each project, the code related to each project be clear.
- Depending on the programming language, please include a README.txt describing how to compile it. For example for python include a requirements.txt file with all packages needed to load from pip. For Javascript please include a package.json with all packages needed.
- If your program will run in Docker, please include the Dockerfile with the instructions to build it in the README.txt file.
- Make sure to pay attention to the scoring rubric to get as many points as possible.
- Most scoring will be completed when you have uploaded your Evidence to the Git Repo.
- If you are unable to upload to a git-repo, please talk to the Tech Chair for alternative options.
- All code, and evidence must be submitted by 3pm on the day of the competition. After you submit your code, let the tech chair know so we can verify we can see and judge it. Missing or incomplete evidence of program execution and testing will result in a 0.

Scoring Rubric

Program 1: CLI : Completeness of Solution	75
Program 1: CLI : Correctness of Output	50
Program 1: CLI : Validation of Input	20
Program 1: CLI : Internal Documentation	35
Program 1: CLI : Quality of Work	35
Program 1: CLI : Efficiency of Code	35
Program 2: API : Completeness	75
Program 2: API : Correctness of Output	50
Program 2: API : Validation of Input	20
Program 2: API : Internal Documentation	35
Program 2: API : Quality of Work	35
Program 2: API : Efficiency of Code	35
Program 3: GUI : Completeness	75
Program 3: GUI : Correctness of Output	50
Program 3: GUI : Validation of Input	20
Program 3: GUI : Internal Documentation	35
Program 3: GUI : Quality of Work	35
Program 3: GUI : Efficiency of Code	35
Interview	100
Online Test	150
Résumé Penalty (0 or -50)	
Clothing Penalty (0 to -20)	

Total Possible Points	1,000

Program #1 – CLI Program

Project: Fibonacci sequence generator (CLI Program)

Project Goal: You will write a command line program to generate Fibonacci sequences on demand. The program will take several arguments, defined below, and will output the corresponding numbers. Definition: According to mathisfun.com: The **Fibonacci Sequence** is the series of numbers: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ... The next number is found by adding up the two numbers before it.

Requirements: You will run the program from a command prompt, and pass in arguments as defined in this help output:

./Fibonacci-gen [-c|--count] 20 –one-line –last-only –numbering –help

Help for Fibonacci generator:

--help : Print this help

--count|-c : Calculate to this many places. IE: 0, 1, 1, 2, 3, 5 would be the result of -c 6

--one-line: Print all the numbers on one line, separated by commas. Without this option, each number in the sequence will be printed on a new line.

--numbering: Preface each number in the sequence with it's placement: IE for "-c 6 –numbering –one-line" you would get this: "1:0, 2:1, 3:1, 4:2, 5:3, 6:5" where the first number is the count and the second is the Fibonacci sequence. Note: this argument should work with all other arguments.

--last-only: Only print the last number in the sequence

Program #1 – CLI Program – page 2

TESTING:

Please run the following tests and record the output.

A) **.**/Fibonacci-gen -c 6 –one-line –numbering
    a. **1:0, 2:1, 3:1, 4:2, 5:3, 6:5**
B) **.**/Fibonacci-gen –help
    a. **Should output the above help**
C) **.**/Fibonacci-gen -c 6 –one-line
    a. **0, 1, 1, 2, 3, 5**
D) **.**/Fibonacci-gen --count 6
    a. **0**
    b. **1**
    c. **1**
    d. **2**
    e. **3**
    f. **5**
E) **.**/Fibonacci-gen --count 15 –last-only
    a. 377
F) **.**/Fibonacci-gen --count 50 –last-only
    a. (figure it out)

When you have completed this program, please commit all of your code to GitHub, as well as a PowerPoint or similar evidence that clearly shows your program running with proper output. Clearly show in the presentation the testing for all the test cases. List your name and Program Name on your Power Point or evidence and commit that into your Git Repo.

Program #2 – API

Project: Fibonacci-gen API (Rest API)

Project Goal: You will write a micro service to serve up the Fibonacci sequence data to a rest API call. The API spec will be defined in requirements. The code should be self contained, and not call out to the Program 1 CLI. You may re-use code from Program 1 (CLI) in the route method that does the calculations, BUT, the math and calculation code need to be fully in the microservice code.

Requirements:

- All responses will be formatted as JSON
- You will call the api with this base url : http://localhost:8080/api/fibonacci/ (The port can be some other port if required)
- The count will always be passed as the only parameter, all other options will be based as query strings. As follows:
- http://localhost:8080/api/fibonacci/:count/?one-line=true&last-only=true&numbering=true&help=true
- Please repeat all the tests from program #1, but the output should be as JSON. So the JSON will look like this

```
{
  "sequence": “0,1,1,2,3,5”
  }
```

Unless the output is not one “one-line” then it should be returned as a JSON array

You may use any tool you wish to call the API endpoint. Curl, Postman, Bruno, etc.

When you have completed this program, please commit all of your code to GitHub, as well as a PowerPoint or similar evidence that clearly shows your program running with proper output. Clearly show in the presentation the testing for all the test cases. List your name and Program Name on your Power Point or evidence and commit that into your Git Repo.

Program #3 (GUI)

Project: Grade Program (GUI)

Project Goal: You will write a program to create a screen which will allow for the entry of grades for 5 different classes. **This must be a GUI program, if it is not built with a GUI (Either Windows popups, webpage, or something graphical) then you will loose 100 points.** Each class will keep track of the grades with an average grade to be shown when selected. Output will also include high score and low score for each class. Number of students included in the calculation.

Requirements:

- Scores of .5 or more will be rounded up to the next whole number for the average score. Example a score of 89.5 will be 90 on the final average.
- Valid score range is from 0 to 100.
- Classes: Programming, Art, Science, Math, and History.

Screen Output will include (For each of the 5 classes):

Class Name:

Scores Entered:

Current Average:

High Score:

Low Score:

Buttons or Menu Choices will include:

Add Grade (If you choose to allow entry via a Menu Item) (optional)

Calculate (This will add the entered values into the summary)

Clear (This will clear the current input, and not include it in calculation)

Reset (This will reset all previous inputs, and allow you to start over)

Print Summary (If you are a Console application and need an option to print the summary) (optional)

Exit

All screens will be clearly labeled with contestant number, program number and title.

All boxes will be clearly labeled.

When you have completed this program, please commit all of your code to GitHub, as well as a PowerPoint or similar evidence that clearly shows your program running with proper output. Clearly show in the presentation the testing for all the test cases. List your name and Program Name on your Power Point or evidence and commit that into your Git Repo.

The following are the expected tests for your code:

Test 1: Enter 5 students scores as follows:

Programming: 90, Art: 75.5, Science: 80.9, Math: 89, History: 81
Programming: 80, Art: 60.5, Science: 81.2, Math: 93, History: 91
Programming: 85, Art: 85.5, Science: 80.3, Math: 67, History: 75
Programming: 90, Art: 82, Science: 80.2, Math: 89, History: 62
Programming: 85, Art: 99, Science: 86.3, Math: 88, History: 66
Print the Summary

Test 2: Enter a negative quantity for any class and have an error generated while not losing the running total

Test 3: Add 2 more students scores:

Programming: 85, Art: 99, Science: 86.3, Math: 88, History: 66 (**But hit clear so this students score is cleared and not included)**
Programming: 99, Art: 60, Science: 79, Math: 81, History: 99
Print the Summary

Final output should show the stats **for 6 Students**.