23S-COM SCI-130-LEC-1 Midterm

CHARLES ZHANG

TOTAL POINTS

75 / 97

QUESTION 1

1 1 **6 / 8**

+ **2 pts** Reproducible state; roll back to previous working version
✓ **+ 2 pts** *Backing up work*
✓ **+ 2 pts** *Documenting progress; revision history*
+ **2 pts** Post-mortems
✓ **+ 2 pts** *Collaboration with team*
+ **2 pts** Legal pedigree
+ **2 pts** Other
+ **0 pts** Allows multiple features to be worked on simultaneously (temporarily scored at 0 points).
+ **0 pts** Memory efficiency / store incremental changes
+ **0 pts** Intentionally set as a 0 point problem to use as a "star" to look back later.
+ **0 pts** Version control == history
+ **0 pts** Uniform version of code? Scalability? Source of truth?

Source control isn't always required to enforce code reviews.

QUESTION 2

2 2 **2 / 4**

+ **1 pts** git checkout change-130
+ **1 pts** git rebase main
+ **1 pts** git switch/checkout main
+ **1 pts** git merge change-130
✓ **+ 4 pts** *Correct final commit graph*
- **1 pts** main pointer not updated
✓ **- 1 pts** *rebased main on top of change-130*
✓ **- 1 pts** *Wrong commands*
- **1 pts** Ran a "commit" based flow

QUESTION 3

3 3.1 Dockerfile **1 / 1**

✓ **+ 1 pts** *Yes*
- **1 pts** No

QUESTION 4

4 3.2 Makefile **1 / 1**

- **1 pts** Yes
✓ **+ 1 pts** *No*

QUESTION 5

5 3.3 Unit test **1 / 1**

✓ **+ 1 pts** *Yes*
- **1 pts** No

QUESTION 6

6 3.4 C++ Object **1 / 1**

- **1 pts** Yes
✓ **+ 1 pts** *No*

QUESTION 7

7 3.5 Personal **1 / 1**

- 1 pts Yes

✓ **+ 1 pts** *No*

QUESTION 8

8 3.6 Team **1 / 1**

✓ **+ 1 pts** *Yes*

- 1 pts No

QUESTION 9

9 4 **6 / 6**

✓ **+ 1 pts** *Correct*

✓ **+ 1 pts** *Hermetic*

✓ **+ 1 pts** *Flexible*

+ 1 pts Easy to use

✓ **+ 1 pts** *Automatable*

✓ **+ 1 pts** *Fast*

✓ **+ 1 pts** *Manages dependencies*

+ 1 pts Integrates with tooling (CR, release, query)

+ 1 pts Scalable

+ 1 pts Efficient

+ 1 pts Other

+ 1 pts Portable

+ 0 pts Zero

QUESTION 10

10 5 **3 / 4**

+ 2 pts Fast to execute

+ 2 pts Fast to write

+ 2 pts Resilient to refactors (not tied to implementation); tests state, not implementation

✓ **+ 2 pts** *Readable / easily verifiable to be correct*

+ 2 pts Tests only a single aspect/feature/property

+ 2 pts Closely follows class API

✓ **+ 1 pts** *Deterministic, i.e. not flaky*

+ 1 pts Comprehensive

+ 1 pts Boundaries : tests edge cases

+ 1 pts Meaningful, tests something of substance

+ 0 pts Automated/automatable -- placeholder

+ 0 pts Intentional 0 points : used as a "star" function to check later.

QUESTION 11

11 6 **2 / 2**

✓ **+ 2 pts** *<= 25% Integration tests*

+ 0 pts `>25% integration tests

+ 0 pts No number

QUESTION 12

12 7 **10 / 12**

✓ **+ 2 pts** *Add unique element, check that size of container increased by 1*

+ 2 pts Add non-unique element, check that the size of the container is unchanged

✓ **+ 2 pts** *Add unique element, check that pair::first points to an element that is equivalent to the one you were adding*

✓ **+ 2 pts** *Add non-unique element, check that pair::first points to an element that is equivalent to the one you were adding*

✓ **+ 2 pts** *Add unique element, check pair::second is true*

✓ **+ 2 pts** *Add non-unique element, check pair::second is false*

+ 2 pts Other

+ 0 pts Check all elements in set are unique after addition

+ **0 pts** Unnecessary tests
+ **0 pts** Underspecified
- **2 pts** Asserted something that goes against spec

QUESTION 13

13 8 **2 / 4**

✓ **- 1 pts** *a selected*
✓ **+ 2 pts** *b selected*
✓ **- 1 pts** *c selected*
✓ **+ 2 pts** *d selected*

QUESTION 14

14 9 **6 / 6**

✓ **+ 2 pts** *a selected*
✓ **+ 2 pts** *b selected*
✓ **+ 2 pts** *c selected*
- **1 pts** d selected
- **1 pts** e selected

QUESTION 15

15 10 **4 / 4**

+ **4 pts** None
✓ **+ 4 pts** *:coverage*
+ **0 pts** :latest
+ **0 pts** Incorrect
- **2 pts** Bad reasoning

QUESTION 16

16 11 **0 / 4**

+ **4 pts** Runtime inputs may result in compile-time variable bounds being too large
+ **4 pts** Problem is NP-complete
✓ **+ 0 pts** *Buffer overrun can only be found at runtime/dynamically allocated buffers can't have size known at compile time*
+ **0 pts** Talk about behavior being unbounded without mentioning variable bound specifically
- **2 pts** Underspecified
- **2 pts** Troublesome reasoning
+ **0 pts** Halting problem
+ **0 pts** Incorrect

QUESTION 17

17 12 **4 / 4**

+ **4 pts** Slow
✓ **+ 4 pts** *Limited by test cases run during the analysis*
+ **0 pts** Incorrect
+ **2 pts** Code readability
- **2 pts** Troublesome reasoning
+ **0 pts** Only focused on resource expense

QUESTION 18

18 13.1 Function Fibonacci **1 / 1**

✓ **+ 1 pts** *none*
+ **1 pts** info
+ **0 pts** warn
+ **0 pts** error
+ **0 pts** fatal
+ **0 pts** blank, no answer

QUESTION 19

19 13.2 Your server **1 / 1**

+ **0 pts** none
✓ **+ 1 pts** *info*
+ **0 pts** warn
+ **0 pts** error

+ 0 pts fatal

+ 0 pts blank

QUESTION 20

20 13.3 A user **1 / 1**

+ 0 pts none

+ 0 pts info

+ 0 pts warn

✓ **+ 1 pts** *error*

+ 0 pts fatal

QUESTION 21

21 13.4 Logging step **0 / 1**

✓ **+ 0 pts** *none*

+ 1 pts info

+ 0 pts warn

+ 0 pts error

+ 0 pts fatal

QUESTION 22

22 13.5 Webserver thread **1 / 1**

+ 0 pts none

+ 0 pts info

✓ **+ 1 pts** *warn*

+ 1 pts error

+ 0 pts fatal

QUESTION 23

23 13.6 Logging progress **1 / 1**

+ 0 pts none

✓ **+ 1 pts** *info*

+ 0 pts warn

+ 0 pts error

+ 0 pts fatal

QUESTION 24

24 13.7 Complex algo **0 / 1**

+ 1 pts none

✓ **+ 0 pts** *info*

+ 0 pts warn

+ 0 pts error

+ 0 pts fatal

+ 0 pts blank

QUESTION 25

25 13.8 Webserver setup **1 / 1**

+ 0 pts none

+ 0 pts info

+ 0 pts warn

+ 0 pts error

✓ **+ 1 pts** *fatal*

QUESTION 26

26 13.9 Virtual method **1 / 1**

+ 1 pts none

+ 0 pts info

+ 1 pts warn

✓ **+ 1 pts** *error*

+ 0 pts fatal

+ 0 pts Blank, not answered

QUESTION 27

27 13.10 User purchase **1 / 1**

✓ **+ 1 pts** *none*

+ 0 pts info

+ 0 pts warn

+ 0 pts error

+ 0 pts fatal

+ 0 pts blank

QUESTION 28

28 13.11 Static file 0 / 1

+ **1 pts** none
+ **1 pts** info
+ **0 pts** warn

✓ **+ 0 pts** *error*

+ **0 pts** fatal

QUESTION 29

29 14.1 Description 2 / 4

✓ **+ 1 pts** *What*

+ **1 pts** Why

✓ **+ 1 pts** *How : must include something like getConfig and parsing to get a point; something other than just describing the function name.*

+ **1 pts** Testing
+ **0 pts** No description
+ **0 pts** Commented on description, didn't actually add a description.
+ **0 pts** No username/UID

QUESTION 30

30 14.2 Comments 14 / 18

✓ **+ 2 pts** *1-4: Use dependency injection*

✓ **+ 2 pts** *7: Bad function name*

+ **2 pts** 8-13: Bad variable names
+ **2 pts** 9-10: Bad magic number literals
+ **2 pts** 13: Unnecessary parentheses
+ **2 pts** 17: Inconsistent brace formatting
+ **2 pts** 19: Missing "this."

✓ **+ 2 pts** *22 : Should do something more than LOG(INFO)*

✓ **+ 2 pts** *25: Don't use fatal log level*

✓ **+ 2 pts** *27: Cannot throw string / Should not throw*

+ **2 pts** 1 other good comment

✓ **+ 4 pts** *2 other good comments*

+ **2 pts** 13 : Flagged return type

UCLA username: charleszhang UID: 305417655

Answer Sheet

(1) 2 words min, 1 sentence max for each reason

1. Preventing loss of progress
2. Promotes collaboration
3. Allows history of project to be viewable
4. Enables code reviews to control what is added to the codebase

(2)

```
git checkout main
git rebase change-170
git push
git branch -d change-170
```

(3) Belongs in source control?

Yes	No	
✓	☐	Dockerfile
☐	✓	Makefile generated by CMake
✓	☐	Unit test .cc file
☐	✓	C++ Object files (.o files)
☐	✓	Personal .bashrc file
✓	☐	Team style config

(4) 1 word min, 1 sentence max for each property

1. Correct
2. Hermetic
3. Flexible
4. Automatable
5. Fast
6. Automatically tracks dependencies

UCLA username: charleszhang UID: 705419659

(5) 2 words min, 1 sentence max for each property

1. Human readable/self-documenting
2. Can be repeated

(6)

1. 5% integration vs. 95% unit

(7) (not all bullets required)

- Increases the container size by the # of unique inserted elements
- Inserts unique elements to the container
- Doesn't insert duplicate elements!
- The first element of the pair is an iterator to the inserted element if the insertion occurred or the existing element if it didn't
- The second element of the pair is set correctly when the element is inserted / when it's not inserted
-
-
-

(8) Circle answer(s): [✓ a] [✓ b] [✓ c] [✓ d]

(9) Circle answer(s): [✓ a] [✓ b] [✓ c] d e

(10) 1-2 sentences only:
You should apply the :coverage tag to this container. Tags are intended to quickly communicate information about a given container, so we should aim to provide an accurate and concise description.

(11) 1 sentence only:
If a buffer is dynamically-sized, it's impossible to determine its bounds during static analysis.

(12) 1 sentence only:
Runtime analysis is restricted to catching bugs discovered by our test coverage, so it can be unable to prove that a bug doesn't exist.

(13) Fill in / mark one box per row:

none	info	warn	error	fatal	
☒	☐	☐	☐	☐	Function Fibonacci(n) wants to ensure that argument n > 0
☐	☒	☐	☐	☐	Your server's main() wants to report the port it serves on
☐	☐	☐	☒	☐	A user wants to update their profile but your server noticed the database is UNREACHABLE
☒	☐	☐	☐	☐	Logging step-by-step progress of a complex algorithm (e.g. leader election in Paxos or Raft)
☐	☐	☒	☐	☐	Webserver thread pool is exhausted, so it can't accept any new requests
☐	☒	☐	☐	☐	Logging progress of webserver setup
☐	☒	☐	☐	☐	A complex algorithm logs information about every update to internal state
☐	☐	☐	☐	☒	Webserver setup hit an unrecoverable failure condition
☐	☐	☐	☒	☐	Virtual method dispatched to a concrete implementation with body having `throw Unimplemented("TODO");`
☒	☐	☐	☐	☐	A user is purchasing CS books and and you want the webserver to record the payment and shipping details
☐	☐	☐	☒	☐	The StaticFileHandler can't find the requested file

UCLA username: charleszhang   UID: 305412659

(14)

New CL Description:
Implemented ConfigStorage() constructor and member function that takes in a user string, fetches the config string of the user using ConfigDatabase, parses the string into a Config object using ConfigParser, and stores a mapping of user -> Config object in ConfigDatabase.

Line	**Comment**
1	Instead of creating these objects inside ConfigStorage(), can we have them be passed in to use dependency injection and make the code more modular/easier to test?
12	This should be an error-level log, not info-level
25	Why is this a fatal log? Shouldn't it be an info-level log? And can we add some identifying info on the user to the log?
16	Is it possible to use a more descriptive type name than std::string here?
27	Can we log an error here and avoid throwing an exception? maybe the function signature can just return a bool indicating success/failure
7	Please rewrite this with a more descriptive function name and more readable implementation if possible.
21	What is being compared here? It doesn't seem possible that a config object could be a null pointer. Should config be a pointer to a Config object?