Course Housekeeping & Final 30 Days Roadmap

JD Kilgallin

CPSC:480

11/09/22

Notes

- A new calendar is provided at the end of this housekeeping slide deck and in the README of the course GitHub repo for easy reference.
- Today's agenda:
    - Announcements
    - Reminders and grade info
    - Project 3 details and updates
    - Exercise 6 review
    - Lecture and quiz schedule
    - Project 4 preview
    - Optional extra credit assignment
    - Final 30 Days Roadmap Calendar
    - SCM main topic lecture:
        - Software build, release, and deployment activities and processes
        - Software Configuration Management (SCM) concepts

Grading Announcements

- Quiz 7 (lectures 18-21) Wednesday. Project 4 will also be announced.
- A take-home practice final will be given Nov 16, due Nov 21 as a quiz.
- There is a new opportunity for extra credit worth 1-2% of final grade.
- Ex. 6 grades posted Saturday. Grade spread was pretty wide. All grades curved up by 14.3% to put the median at B-. See me after class/at office hours for individual grade details.
- Participation bonus grades were updated last week; expect one more update after Thanksgiving, and final grade determined at very end. I score this biweekly out of 5, take your average over each period, double, and round up. Basically 5 periods down, 3 to go including the current one.
- Reminder that late work is not accepted without a documented, valid excuse; don't leave your work to the last minute.
- There will be a new form for post-project surveys. Use it for projects 3 & 4.

Other Announcements

- Wednesday, Nov 16, will be guest speaker Mikyla Wilfred, Keyfactor's product owner and UI/UX Engineer, and 2021 employee of the year.
- Office hours are now officially 6:30-7:30, and begin in CAS 134, then 230.
- There will be no regular office hours Wednesday, Nov 23 (the day before Thanksgiving); office hours by appointment only that week.
- An extra office hour/final review/guided study session will be held during regular class time Monday, Dec 5, in my office (CAS 230).
- SCM and Automation lectures this week are swapped from syllabus plan.
- 3 more fully examinable lectures after today: automation, risk, other skills
- There may be additional examinable content in *slides* for lecture 24 (UX), 26 (Python/ Thanksgiving), and 28 (Conclusion); all slides are examinable.
- Project repos will not be renamed from "…_P2" in order to preserve URLs.

Project 3

- Checkpoint Wednesday; early submission for whole project Saturday (2%) or Sunday (1%); final deadline Monday, Nov 14, 1 week from today; team participation survey next Tuesday, Nov 15.
- Your code review requests should all be out by now in order to approve and merge branches by Wednesday.
- You may use any debugging or testing tool (e.g. Visual Studio) to analyze code coverage, or you may **estimate** coverage.
- Gold standard without an automated tool would be to add trace logging after every conditional (similar to TESTBRANCH statements in exercise 6), and analyze the log to confirm that every possible message appears.
- At minimum, document or add comments explaining how different branches get tested. Inadequate justification may not earn full credit.
- The standard will also be relaxed to 50% coverage (grade calculated with 2*coverage instead of 1x, up to a max of 100% - no extra credit for higher). Pass rate should still be 100%. Formally, min(2*coverage,100%)*passRate.

Exercise 6 Grading

- Code coverage and pass rate were each calculated as the minimum of the output on *your* set of tests, and your set and mine *combined*. E.g. if you had 4 tests and got a 100% pass rate and 95% coverage, and I added 14 tests and got 100% coverage but 6 failed, your calculated pass-rate*coverage grade would be (12/18=66.67%)*95%=63.33% (12.67 out of 20% component).
- There were 4 bugs in 30% component; ½ points for finding & ½ for fix quality.
- At least 5 tests are required to verify all input cases, and you could reasonably write up to 16 to test pass/fail scenarios for each edge case or functional requirement. 40% component scored 75% on addressing 10 specific cases and 25% on number of tests – 5 points per test up to 5 tests.
- kilgallin/SWEF22-Exercise-6 is a fork of JDKeyfactor/SWEF22-Exercise-6; PR should have gone from your branch to main in kilgallin, not JDKeyfactor. NBD.

Exercise 6 Bugs

- There were 4 bugs; 1 in twoSumNaive() and 3 in twoSumwSort():
- Easy: twoSumwSort never set possible="true"
- Medium: Both functions would return true even if i==j (i.e. if one of the elements in the array is half the target, it will say that *is* possible, when the spec requires them to be distinct elements).
    - The best fix for this is to start the inner loop/binary search at i+1 instead of 0. This also cuts the worst case time in half; if it checks (i,j)=(1,4), it doesn't later check (4,1)
- Hard: twoSumwSort returned indexes of the *sorted* array. To identify this required a test case with the input *out of order,* such that the return indexes are not in their original spot. The fix is to make a copy of the input array within twoSumwSort, sort the copy, then do find on the original input
- Since twoSumwSort needed a significant rewrite to fix the three bugs, it was a good idea to fix the return object to match the other functions (no result struct declared until the return statement).

Exercise 6 Complexity

- Cyclomatic complexity is a small, positive number. Runtime complexity is a function of the input size, e.g. "O(n^2)".
- Cyclomatic complexity counts the number of distinct paths or branches in a function, and starts at 1 by definition. It goes up by 1 for each *condition* that determines the next code to be executed. Short-circuit evaluation means that in the line "if(a || b){…}", if a is true, b *doesn't get evaluated*. Equivalent to "if(a){if(b){…}}". CC+=2.
- twoSumNaive – runtime O(n^2): double loop. CC 4 "***for***(){***for***(){***if***(){…}}}"
- twoSumwSort – runtime O(nlogn): binary_search is $\log_2(n)$, in a loop makes $n*\log_2(n)$ (extra search *adds* 1*n). CC 3 "***for***(){…{***if***(){…}}}".
- twoSumwHash – runtime O(n): single loop. CC 4 "***for***(){***if***(_**&&**_){…}}"
- Adding an "i != j" check on twoSumNaive or twoSumwSort adds 1 CC.

Exercise 6 Tests

- Edge cases (extreme input values) to look out for include: minimum target (0), very large target, minimal array (empty), & very large array.
- The corner case of 0 target, empty array is especially important.
- Since one method required sorting, an unsorted input is important.
- The spec required *distinct* i,j, so it's important to have an input where the input contains half the target and no other solutions. But repeat elements are allowed, so input with ½ the target twice should pass (e.g. t=4, e={2, 2})
- The spec states that no integer in elements should be larger than the target, but the code should still behave on this case (security issue).
- Basic code coverage required at least one positive test (expected =true) and one negative test. Proper verification should have positive/ negative for each special case.
- Tests can include more code than static variable assignment. This was a good candidate for fuzz testing.

Meeting schedule *(*fully examinable)*

- *21* – Today (SCM)*
- *22* – Wednesday, Nov 9 (Automation)*
- *23* – Monday, Nov 14 (Risk)*
- 24 – Wednesday, Nov 16 (UI/UX/product ownership guest lecture)
- *25* – Monday, Nov 21 (Research, sales, marketing, support, sysadmin, etc)*
- 26 – Wednesday, Nov 23 (Python for interviews, automation, data science)
- 27 – Monday, Nov 28 (final project presentations)
- 28 – Wednesday, Nov 30 (conclusion and final review)
- ** – Monday, Dec 5 (bonus office hour)
- ** – Wednesday, Dec 7 (final)

Final grade breakdown

- Completed categories so far are 50% of final grade: midterm (20%), project 1 & 2 (10%x2), and exercises (10%, lowest dropped). Other 50% of grade still to go.
- Project 3 this week (checkpoint Wednesday, project Monday) – 10%.
- Project 4 from this week to end of classes – 10%.
- Final Dec 7, 5:15-7:15, CAS 134 – 20%.
- Quizzes – 10%, lowest dropped:
  - 1-6 – Already completed
  - 7 – Wednesday, Nov 8 (lectures 18-21)
  - 8* – Tuesday, Nov 15 (project 3 survey)
  - 9 – Monday, Nov 21 (take-home practice final)
  - 10– Monday, Nov 21 (lectures 22-24)
  - 11* – Monday, Nov 28 (peer presentation evaluations)
  - 12* – Tuesday, Nov 29 (presentation own-team evaluation)
  - 13* – Wednesday, Nov 30 (course feedback survey)
  - 14* – Friday, Dec 2 (project 4 survey)
  - **graded for completion only*

Extra Credit

- Pick an appropriate Software Engineering topic not covered in lecture and prepare a 5-15-minute presentation on it (about 3-10 of my slides).
- Submit by forking the kilgallin/SWEF22 course repository, and issuing a pull request with a .pptx and .pdf of the slides in the new "supplemental" folder, with a filename matching the topic. Plan to present the topic to the instructor (and optionally interested students) before or after a remaining class, or at Office Hr. Last date for presentations is OH Wednesday, Nov 30.
- An appropriate, valid topic submission will be 1% of final grade extra credit. This will also apply toward class participation/improvement grade (up to another 1% of final grade). 1 topic per student; no duplicate topics.
- Example topics include: mobile/game/cloud/IoT development; databases; AI; specific frameworks, technologies, languages, applications, or historical events; OSS; performance/scale testing; cryptography or security testing; data mining; emerging trends; integrations/APIs; CRM; team management.

Project 4 Preview

- High-level: Conduct another sprint and present to stakeholders (me + rest of class).
- Hold a retrospective of previous sprint (Projects 2 3); identify top process issues.
- Identify remaining work to produce a bare-minimum viable prototype. If it cannot reasonably be completed in two weeks, evaluate what else can be taken out. Your goal is to follow the development process, not to release functional software. Stakeholders would just want to see that some progress has been made and whether the project schedule is on track.
- Develop a plan to address issues, complete sprint, estimate "future" work.
- Assign remaining work to build "something" to show stakeholders. You may designate 1-2 primary engineers to do this. **Checkpoint 1 Tuesday, Nov 22.**
- Perform project management/software release activities e.g. product documentation, risk analysis, product metrics, change control. **Checkpoint 2 Sun, Nov 27.**
- Present your prototype & progress notes as if I'm considering assigning the project to another team, and the rest of the class might join. **Presentations Monday, Nov 28.**
- Report on final product, development process, and what you learned. **Friday, Dec 2.**

6 Sun	**7 Mon** **L21 SCM***	8 Tue	**9 Wed** **L22 Autmtn*** **Project 3 checkpt** Project 4 asgned. **Quiz 7 (L18-21)**	10 Thur *Q7 grades;* *P3 chkpt grade?*	11 Fri	12 Sat Project 3 2% EC
13 Sun Project 3 1% EC	**14 Mon** **L23 Risk*** **Project 3 due**	15 Tue Quiz8 (P3 survey)	**16 Wed** **L24 UI/UX** *Mikyla guest spkr* Q9:takehome out	17 Thur *Q8, P3 grade?*	18 Fri	19 Sat
20 Sun	**21 Mon** **L25 Skills*** **Q9:takehome due** **Quiz 10 (L22-24)**	22 Tue **Project 4 chkpt 1**	**23 Wed** **L26 Python** **No OH** *Python language is not examinable*	24 Thur *Thanksgiving; Q9-10, P4c1 grade?*	25 Fri	26 Sat
27 Sun **Project 4 chkpt 2** *Participation bonus update?*	**28 Mon** **L27 Present-ations** **10-min P4 pres.** Q11 pres eval	29 Tue Q12 pres selfeval *P4 chkpt2 grade?*	**30 Wed** **L28 Concl.** *Incl final review* Q13 class survey	Dec 1 Thur Project 4 2% EC	2 Fri P4 1% EC @6 PM **Project 4 due** Q14 (P4 survey)	3 Sat *Q11-14 grade?*
4 Sun *Project 4 grade?*	5 Mon Bonus office hour /Extra final review normal class time my office CAS230	6 Tue	**7 Wed** **Final Exam** Dec 7, 5:15-7:15 CAS 134 (same)	8 Thur	9 Fri *Final exam/ semester grade?*	10 Sat

Class days **bold** on top| Due dates for graded work **bold** on bottom|*=full lecture examinable| *italics* are fyi only| Some dates subject to change