CS 61B Data Structures, Spring 2019

Instructor: Josh Hug
Lecture: MWF 3-4 PM, Wheeler 150

General Background Information

Welcome to CS 61B

The CS 61 series is an introduction to computer science, with particular emphasis on software and machines from a programmer's point of view. CS 61A covered high-level approaches to problem-solving, providing you with a variety of ways to organize solutions to programming problems as compositions of functions, collections of objects, or sets of rules. In CS 61B, we move to a somewhat more detailed (and to some extent, more basic) level of programming.

In 61A, the *correctness* of a program was our primary goal. In CS61B, we're concerned also with *engineering*. An engineer, it is said, is someone who can do for a dime what any fool can do for a dollar. Much of 61B will be concerned with the tradeoffs in time and memory for a variety of methods for structuring data. We'll also be concerned with the engineering knowledge and skills needed to build and maintain moderately large programs.

Background Knowledge

This class assumes you have taken CS61A, CS88, or E7, or have equivalent background to a student who has taken one of these courses. The course is largely built upon the assumption that you have taken CS61A or CS88, and E7 students may find the beginning of the course to be a bit scarier, particularly when it comes to object oriented programming.

We assume you are coming in with zero Java experience. Nonetheless, we will move through basic Java syntax very quickly. Though the syntaxes of Java, Python, MATLAB, Scheme, etc. are enormously different, the underlying computational models are surprisingly similar.

If you already have Java experience, great! We hope that you'll help out your fellow students in discussion, lab, and on Piazza, particularly in the opening weeks when everyone is catching up on Java.

Knowledge of UNIX is not required, though you might find it useful if you want to use the lab computers instead of your own computer. Lab 1 will give you resources to learn UNIX if you end up deciding use the lab computers instead of your own.

Is this the right course for me?

This is a course about data structures and programming methods. It happens to also teach Java, since it is hard to teach programming without a language. However, it is not intended as an exhaustive course on Java, creating Android apps, user interfaces, graphics, or any of that fun stuff.

Some of you may have already had a data structures course, and simply want to learn Java or C++. For you, self-study may be a better option. CS 9F (C++ for programmers) and CS 9G (Java for programmers) are both one-unit self-paced courses that will teach you more of what you want to know in less time. There is no enrollment limit for that course, and you work through it at your own pace after the first and only lecture.

Finally, the 1-unit self-paced course CS 47B is for students "with sufficient partial credit in 61B," allowing them (with instructor's permission) to complete the 61B course requirement without taking the full course.

Discussion and Lab Sections

Each week there is a 1 hour discussion section and a 2 hour lab section headed by a GSI and supported by volunteer academic interns. Information about the staff running each section can be found on the staff page.

Discussion and lab attendance are not usually mandatory, but attendance earns you 2 pacing points, which are described in more detail below.

There are 3 labs for which attendance is mandatory for full credit: Lab 1, Lab 4 (project 1 code review lab), and the Lab 14 (project 2 checkoff lab).

There will be three types of discussion sections and two types of lab sections. The discussion section types are:

- LOST: Special drop-in sections. Assumes no prior knowledge. A safe space for those of you who are feeling behind.
- Standard: Assumes exposure to lecture material. Reinforcement of class fundamentals.
- Exam prep: Assumes solid understanding. Hard exam-level (and beyond) problems.

The two lab types are:

- Standard: Implementation of basic data structures.
- Challenge (weeks 5-13 only): Very tricky programming problems.

Labs are worth part of the course grade. Points are earned by submitting working code to gradescope. For each week, you can earn points for either Standard or Challenge labs (but not both). Lab materials are available online, so you will be able to do much of the work ahead of time. Note that challenge labs are new and will probably be released without much time before each week's lab meeting.

Lab, Discussion, and Mentor GSIs

We will lab and discussion section signups shortly before classes start. Details TBA.

In addition, you may also choose a mentor GSI. Your mentor GSI does not necessarily have to be your lab or discussion GSI. If you do not choose a mentor GSI, one will be assigned for you randomly. Details TBA.

Your mentor GSI will keep an eye out for you, and will also be your point-of-contact for issues when you need help or advice.

Online Resources

The course home page will provide one-stop shopping for course information. The course schedule as well as all handouts, homework, labs, FAQs, etc., will be posted there.

Our discussion forum this semester will be Piazza. For most questions about the course, Piazza is the right place to ask them. The course staff read it regularly, so you will get a quick answer. Furthermore, by posting online as opposed to emailing us directly, other students benefit by seeing the question and the answer. Don't forget to check Piazza before asking your question, just in case someone else has already posted it.

The e-mail address cs61b (at) berkeley.edu will send a message to the course staff (Josh and the head TAs). You can use it for correspondence that you don't want to send to Piazza. The head TAs and Josh all read it, so you will usually get a quick reply. If you send a question that is of general interest, we may post the response on Piazza (we will keep personal information out of it, of course). If you have any problems that require an exception to course policy (e.g. medical emergencies or sudden necessary travel that result in extended absences), please contact cs61b (at) berkeley.edu. **Please do not email Josh for exceptions. Email cs61b (at) berkeley.edu**.

To talk with us, the best way is to come during regular office hours (posted on the home page). Many of us are available at other times by appointment. Please don't be shy. Web pages, email, and Piazza are useful, but it's still much easier to understand something when you can talk about it face-to-face. Even if you're an "online section" (or W61B) student, you're still welcome to come to office hours if you're in town. Office hours are concentrated Monday to Wednesday because we hold labs all day Thursday and Friday.

Course Materials

There is no required textbook for the class.

I've written our own course notes for the first four weeks of the course. These should be sufficient for you to understand what we need to know about Java. They can be found at https://joshhug.gitbooks.io/obscurantism-in-java-first-edition/content.

If you find these notes insufficient, you might consider consulting Paul Hilfinger's (free) Java Reference or Head First Java, 2nd Edition by Sierra and Bates (O'Reilly, 2005). These are not required for the course.

The optional textbook for the weeks 5-14 of the course is Algorithms, 4th Edition by Wayne and Sedgewick.

All textbooks for this course are optional. Homework will not be assigned from them, and alternate readings will be provided when possible.

The official description of the Java core language is available online in The Java Language Specification (Java SE 11 Edition) by James Gosling, Bill Joy, Guy Steele, Gilad Bracha, Alex Buckley, and Daniel Smith. It's extremely thorough and easy to read (once you understand how to read it).

Software

This official coding environment and text editor for the course is the Integrated Development Environment (IDE) called IntelliJ, though we will not introduce this IDE until Week 2. At your own discretion, you may instead use Vim, Emacs, Sublime, or IDEs like Eclipse, Netbeans, Emacs etc. Whatever you use, however, your submitted solutions must conform to our expected style guide. We strongly recommend that you use IntelliJ starting as soon as you finish project 0. We will not officially support any editing / programming environment other than IntelliJ.

This semester, we will use Java 11. You may use either the open source or closed source version of the JDK (see lab 1 setup).

You will be able to do any work you'd like on any Windows, Mac OS X, or Linux computer. You may also remotely log into the instructional machines (which you will receive an account for during the first week), though you should be able to most everything in the course by working natively on your own computer. Information for setting up your own computer is linked in Lab 1b.

We'll be using the version-control system Git this semester. Version-control systems allow you maintain a series of "snapshots" of your files at various points in their development. Used properly, this provides you some back-up protection, so that you can recover previous states of your work when something goes wrong. Also for team-oriented projects (as well as in the real world), version-control systems help manage collaborative work.

You will be learning and using Git for this course to track your work and submit your assignment. In addition to the advantages above, using Git will allow the staff to track your progress in the course and maybe even help you out when you're stuck on bugs. The first lab will teach you the basics of what you will need to know. Feel free to also read official Git documentation.

Expected Coursework

There are five required aspects of the course for which you earn points:

1. Weekly Check-in Surveys
2. HWs
3. Lab Assignments
4. Projects
5. Exams

In addition, you may also earn extra credit by completing additional surveys and by completing projects 2 and 3 early.

Weekly Check-in Survey

While lecture attendance is not required, nor even expected, we do expect you to stay up to date with lecture material. To help keep you on track, there will be 14 weekly check-in surveys due on Sundays at 11:59 PM. These surveys will also be used to keep track of attendance for discussion and lab.

Each weekly check-in survey is worth 8 points, for a total of 80 points. Only your top 10 weekly surveys are counted (out of 14). No late surveys will be accepted.

HW and Lab Assignments

There are 14 weeks of lab in the course, as well as 4 required homeworks.

During Phase I of the course (Weeks 1 through 4), labs will provide you with help getting your computer set up and teach you how to use essential Java programming tools.

During Phases II and III of the course (Weeks 5 - 14), standard labs will usually involve implementation of some data structure or algorithm described in lecture, whereas challenge labs will involve solution of tricky programming problems related to recent material. During such weeks, you can do either (but not both) of the labs for credit.

All standard labs should take no more than two hours to complete, though some may run slightly longer. Challenge labs are an experimental part of the course, and these labs may take much longer. HWs will vary from 3 to 10 hours of work. You will turn in everything electronically using GitHub, and your results will be available on Gradescope. All homeworks and labs are submitted individually (without partners), but you may work with other students subject to the collaboration rules described below.

There is no lab to complete during weeks 10 (exam review) and 14 (Project 3 checkoff).

Homework will be graded on a rigorous suite of correctness tests while labs will receive full credit for "reasonable effort," as evaluated by a small number of relatively simple correctness tests. Passing all tests on Gradescope for homework or labs will ensure full credit as there are no hidden tests. Each of the 12 labs will be worth 16 points (for a total of 192 points), and each homework will be worth 80 points (for a total of 320 points). No homeworks or labs will be dropped.

Programming Projects

In addition to the HWs and labs, there will be 4 programming projects. In these projects you will build an entire system. For project 0, you may optionally work with a partner. For project 3 you will be required to work with a partner, unless you specifically request otherwise (details TBA). Projects 1 and 2 are to be completed independently.

Project 0 and 1 will be relatively easier than projects 2 and 3, taking less time and with greater le scaffolding. Project 2 will be a difficult project (on par with what you might expect from a Hilfinge

projects), though it will be spread over a long period of time. Project 3 will be an even more difficult capstone project in which you will design a project from scratch.

Each project has a specific theme:

- Project 0 (NBody): Introduction to Java
- Project 1 (Deques): Basic Design, Testing, and Code Review
- Project 2 (BearMaps): Testing and Large Scale Implementation
- Project 3 (BYOW): Large Scale Design

For projects 0 and 1, we will ultimately release all tests that determine your grade. In other words, passing all tests on Gradescope will allow you to earn full points for the autograded portion of the project.

Project 2A and 2B will have no publicly available tests. It will be up to you to verify the correctness of your implementation. All tests for Project 2C will be available.

Project 3 will have an autograded portion and a checkoff portion. All tests will be released for the autograder portion.

Projects 0, 1, 2, and 3 will be worth 100, 160, 300, and 400 points respectively. For each project, there are opportunities for "gold points" described in each project specification.

Project 1 will be broken into 2 parts:

- 100 points (project 1A)
- 60 points (project 1B)

Project 2 will be broken into 3 parts:

- 120 points (project 2A)
- 100 points (project 2B)
- 80 points (project 2C)

Edit (5/2/2019): Due to a misconfiguration with the autograder, there is an alternate weighting for project 2:

- 50 points (project 2A)
- 100 points (project 2B)
- 150 points (project 2C)

You will automatically be given the better of these two weightings.

Project 3 will be broken into 5 parts:

- 100 points (phase 1)
- 100 points (phase 2)
- 20 points (partner review #1)
- 20 points (partner review #2)
- 160 points (lab demo)

Completing projects 2 and 3 early will earn you extra credit, as described below.

Exams

There will be two evening midterms on February 20th from 8 - 10 PM and on April 5th from 8 - 10 PM. There will also be a three hour final exam Wednesday May 15th at 7 PM. Midterm 1 will be worth 320 points, midterm 2 will be worth 480 points, and the final exam will be worth 800 points.

You will be allowed to bring one letter size page of handwritten notes (front and back) to the first midterm, two to the second midterm, and three to the final. You will not be required to turn in these sheets, and you may reuse them from exam to exam.

Inspired by the great Paul Hilfinger tradition, exam questions may cover any material whatsoever. For fear of our lives, exams will almost exclusively test material covered in the course.

There will be **no alternate midterm exams**. If you miss an exam, your score will be reweighted with your performance on other exams (see "supersession" elsewhere in this document). Students with disabilities that require alternate exam timing will be honored, so long as you can make a time that overlaps the official time. If you have a disability that prevents your ability to make such a time, we will discuss alternate arrangements with you directly. If you are traveling on official UC business, and have a proctor available, we will allow remote exams to be taken at the same time as the official exam.

Alternate finals will be given in case of a direct final conflict only. If you miss a final exam due to illness, you will be given an incomplete grade in the course and will have to complete the final exam during a future semester.

We release grades for exams on Gradescope. If you believe we have misgraded an exam, request a regrade on the same site with a note explaining your complaint. You should check the online solutions first to make sure that this regrade will make your total score go up as it is possible to lose points from a regrade request.

Due to the university level change in drop deadlines, Midterm 1 grades will be unavailable before the drop deadline. If you're a prospective CS major and you are worried about dropping the course in the time before the drop deadline, we will provide advising sessions to help you decide what to do. Details will be announced at that time.

Exam Supersession

For those of you who miss an exam, have a bad night, or make major improvements over the semester, the exam supersession policy gives you a chance to replace one of your midterm exams.

Specifically, if it helps your score, we will replace one of your midterm scores by its "final statistic equivalent" (FSE). We compute the FSE of an exam as follows:

Let F be the number of standard deviations above the mean that you score on the final. For example, if you are 0.3 standard deviations below the mean, $F = -0.3$. Let $\overline{M}$ be the class-wide mean (not including zeroes) on an midterm. Let σ_M be the class-wide standard deviations (not including zeroes) on a midterm. Your FSE for that exam is $\sigma_M F + \overline{M}$. The FSE cannot go above the maximum possible score for that exam.

If one of your FSEs is better than your original midterm score, we will use the FSE instead. If both are better (e.g. you do much better on the final than either midterm), then we will replace the exam that gives you a bigger overall benefit. If both of your FSE are worse, nothing happens (i.e. doing badly on the final won't hurt your earlier exam scores).

In pseudocode, supersession works as follows:

```
F = (your_final_score - final_mean) / final_stddev

FSE_m1 = m1_stddev * F + m1_mean
FSE_m2 = m2_stddev * F + m2_mean

score_with_m1_replaced = FSE_m1 + your_m2_score + your_final_score
score_with_m2_replaced = your_m1_score + FSE_m2 + your_final_score
score_with_no_replacements = your_m1_score + your_m2_score + your_final_score

your_total_exam_score = max(score_with_m1_replaced, score_with_m2_replaced, score_with_n
```

Extra Credit

There will be a total of 74 points of extra credit in the course:

1. 12 points each for projects 2 and 3. Details in project specs.
2. 8 points for the pre-semester survey (released 1/20/2019).
3. 8 points for mid-semester survey.
4. 16 points for taking the staff created end-of-semester survey.
5. 8 points for taking the official university end-of-semester survey.
6. Up to 10 points for taking Professor Michael Clancy's concept inventory. Details about this calculation can be found on here.

Grades

Your letter grade will be determined by the total points out of the possible 3,152. In other words, there is *no curving in this course*, other than the supersession policy above. Your grade will depend solely on how well you do, and not on how well everyone else does. Unlike other lower division CS courses, the grading bins for 61B generally do not get tweaked at the end of the semester.

Category	Percentage	Points
Homework/Labs	~16.2%	512

Category	Percentage	Points
Surveys	~2.5%	80
Projects	~30.5%	960
Midterms	~25.4%	800
Final Exam	~25.4%	800
Total	100%	3,152

A+	A	A-	B+	B	B-	C+	C	C-	D+	D	D-	F
3070	2900	2710	2450	2280	2150	2000	1850	1488	1292	1088	800	0

These bins were designed to comply with departmental guidelines that the average GPA for a lower-division required course be in the range 2.8 - 3.3, not including students who drop or take the class for a P/NP grade. The design process involved setting of specific standards I expect students to achieve for the A, B, and C bins, with numbers adjusted and other bins interpolated based on a model that I built of predicted student performance. At the end of the semester, we might make the bin boundaries slightly friendlier, though as noted above, unlike other lower division CS courses, I don't typically move them very much, if at all.

We will grant grades of Incomplete *only* for dire medical or personal emergencies that cause you to miss the final, and only if your work up to that point has been satisfactory. Do *not* try to get an incomplete simply as a way to have more time to study or do a project. That is contrary to University policy.

[bins subject to change until the end of the first week of class]

Gold Points Boosting

Gold points can be acquired by completing the 'stretch' goals on the projects. Gold points act sort of like extra credit, though the lower your exam scores, the more your gold points will count.

If you earn g gold points and T points on all exams (after taking into account exam supersession), then your total score in the course will be boosted by $2\left(g - g \cdot \frac{T}{1600}\right)$.

For example, if you earn 8 gold points on project 2 and have 960 points on your exams including exam supersession, then you'll earn a gold boost of $2\left(8 - 8 \cdot \frac{960}{1600}\right) = 6.4$ to your total score in the class.

If we decide a project is too difficult, we reserve the right to move some part of the project into the gold parts section. In such a case, we may readjust the distribution of gold points.

Pacing Points Boosting

There are three ways to earn “pacing” points, which are given for keeping up with the class.

1. Attending discussion in person.
2. Attending lab in person.
3. Attending live lecture or watching the webcast lecture on time.

For each discussion or lab you earn 2 pacing points. For each lecture, you earn 1 pacing point. You may earn up to a maximum of 60 pacing points. For example, one way to achieve the maximum number of pacing points would be: attending 10 discussions in person, 10 labs in person, and keeping up with 20 lectures.

Pacing points use the same formula as gold points, but cannot increase your score beyond 2200. For example, if you earn 60 pacing points throughout the semester and have 600 points on your exams including supersession, then you'll earn a pacing boost of $2\left(60 - 60 \cdot \frac{600}{1600}\right) = 75$ points on your total score. However, if this pushes your total score above 2200, your score will be set equal to 2200.

Policies on Collaboration, Cheating, and Lateness

Deadlines can be stressful, and we know that under extreme pressure, it becomes tempting to start rationalizing actions that you would otherwise yourself consider inappropriate. Perhaps you'll find yourself facing a 61B project deadline, and under all this stress you'll convince yourself that you're just going to cheat for the moment so you can get the points, and that you'll come back later and really learn the thing you were supposed to have learned in order to restore your karmic balance (I've heard something along these lines a few times).

This is a terrible idea. Obviously it's important to learn how to deal with deadlines, but far more important than that, giving into this sort of pressure under not-so-dire circumstances is going to do some damage to your moral compass. Someday, when the consequences are higher than potentially losing a 1/3rd of a letter grade, you may find yourself committing dishonest acts at the cost of someone else's livelihood or life.

Plagiarism on any hw, lab or project will result in a score of -200 on that assignment, which will likely reduce your letter grade by several bins. A second instance of plagiarism on a homework, lab, or project will result in an F in the course. All incidents of plagiarism will be referred to the Office of Student Conduct, including carelessly leaving code up on GitHub. Given our friendly lateness policy (see below), there will be no exceptions to this rule.

During the Spring 2017 semester, we compiled a series of incident reports written by students who were caught plagiarizing. If you find yourself tempted to cheat, you might turn to the words of others who have made the wrong choice for guidance.

HW and Lab Collaboration Policy

In CS61B, we have three types of assignments: homeworks, labs, and projects. The entire point of homeworks and labs is to learn. For homeworks or labs, you should feel free to collaborate with others however you choose, though keep in mind that greater independence is likely to give you a better learning experience (as long as you aren't totally stuck). **Even though we will allow close collaborations on HWs and labs, your code should still be your own work!** Identical or near identical submissions will be treated as plagiarism.

Project Collaboration Policy

By contrast, the projects were designed not just for learning (particularly how to be self-reliant in the context of large unfamiliar systems), but also for the dual purpose of evaluating of your mastery of the course material. As such, they are intended to be completed primarily on your own (or with your partner for the first two projects), particularly when it comes to writing the actual code.

For exams, we will be absolutely unforgiving. Any incident will result in a failing grade for the course, though Berkeley will let you retake 61B next semester. As above, all incidents of cheating will be referred to the Office of Student Conduct.

What constitutes cheating? **The golden rule of academic dishonesty is that you should not claim to be responsible for work that is not yours.**

This is obviously open to some interpretation, and you'll be getting some help from instructors, the internet, other students, and more throughout the course. This is OK, and we hope that the class is an open, welcoming, collaborative environment where we can help each other build the highest possible understanding of the course material. To help (but not entirely define) the bounds of acceptable behavior, we have three important rules for projects:

1. **By You Alone**: All project code that you submit (other than skeleton code) should be written by you (and if applicable, your project 0 or project 2 partner) alone, except for small snippets that solve tiny subproblems (examples in the Permitted section below).

2. **Do Not Possess or Share Code**: Before you've submitted your final work for a project, you should never be in possession of solution code that you (or your partner) did not write. You will be equally culpable if you distribute such code to other students or future students of 61B (within reason). DO NOT GIVE ANYONE YOUR CODE – EVEN IF THEY ARE DESPERATELY ASKING. DO NOT POST SOLUTIONS TO PROJECTS ONLINE (on GitHub or anywhere else)! If you're not sure what you're doing is OK, please ask.

3. **Cite Your Sources**: When you receive significant assistance on a project from someone else, you should cite that assistance somewhere in your source code with the `@source` tag as described in lab 1. We leave it to you to decide what constitutes 'significant'.

For clarity, examples of specific activities are listed below:

Permitted:

- Discussion of approaches for solving a problem.
- Giving away or receiving significant conceptual ideas towards a problem solution. Such help should be cited as comments in your code. For the sake of other's learning experience, we ask that you try not to give away anything juicy, and instead try to lead people to such solutions.
- Discussion of specific syntax issues and bugs in your code.
- Using small snippets of code that you find online for solving tiny problems (e.g. googling "uppercase string java" may lead you to some sample code that you copy and paste into your solution). Such usages should be cited as comments in your hw, lab, and especially project code!

Permitted with **Extreme Caution**:

- Looking at someone else's project code to assist with debugging. Typing or dictacting code into someone else's computer is a violation of the "By You Alone" rule.
- Looking at someone else's project code to understand a particular idea or part of a project. This is strongly discouraged due to the danger of plagiarism, but not absolutely forbidden. We are very serious about the "By You Alone" rule!
- Working on a project alongside another person or group of people. Your code should not substantially resemble anyone else's!

Absolutely Forbidden:

- Possessing another student's project code in any form before a final deadline, be it electronic or on paper. This includes the situation where you're trying to help someone debug. Distributing such code is equally forbidden.
- Possessing project solution code that you did not write yourself (from online (e.g. GitHub), staff solution code found somewhere on a server it should not have been, etc.) before a final deadline. Distributing such code is equally forbidden.
- Posting solution code to any assignment in a public place (e.g. a public git repository, mediafire, etched into stones above the Mediterranean, etc). This applies even after the semester is over.
- Working in lock-step with other students. Your workflow should not involve a group of people identifying, tackling, and effectively identically solving a sequence of subproblems.

We have advanced cheating detection software, and we will routinely run this code to detect cheating. Every semester, we catch and penalize a significant number of people (100+ cases in Spring 2017). Do not be one of them. If you find yourself at such a point of total desperation that cheating begins to look attractive, contact one of the instructors and we can maybe help somehow. Likewise, if 61B is causing massive disruption to your personal life, please contact us directly.

If you admit guilt to an act of plagiarism before we catch you, you will be given zero points on that assignment, and we will not refer your case to the university administration.

Obviously, the expressive power of Java is a subset of the English language. And yes, you can obviously obey the letter of this entire policy while completely violating its spirit. However, this policy is not a game to be defeated, and such circumventions will be seen as plagiarism.

Lateness

The expected deadlines for each assignment are posted on the class website. It is possible to submit work late by requesting an extension token for the assignment. For the occasionally slightly late assignment, you'll be able to use an automated system to get an extension token. The system may reject your request for 3 possible reasons:

1. You've requested too much extension time over the course of the semester.
2. The extension you requsted is too long.
3. Your extension was submitted after the deadline.

If your request is denied, you must contact your mentor GSI to request an extension. Retroactive extensions are possible but will be given sparingly.

We have been intentionally vague about the automated extension system. You should think of the deadlines as the actual deadlines, and extensions as something exceptional that you should use only when necessary.

We are willing to work with you to accomodate your needs if you start falling behind. Under no circumstances should you resort to plagiarism. Because our lateness policy is so accommodating, we will be absolutely strict about pursuing penalties for cheating!

You can find the automated extension system here.

Acknowledgements

Some course handout material derived froms Paul Hilfinger's CS61B handout.

Last built: 2019-05-25 01:26 UTC