LATE

minutes remaining

Hide Time

Manual Save

Question 1. Word Bank Matching (1 point each, 14 points total)

For each statement below, input the letter of the term that is *best* described. Note that you can click each word (cell) to mark it off. Each word is used at most once.

A. — A/B Testing	B. — Agile Development	C. — Alpha Testing	D. — Beta Testing
E. — Competent Programmer Hypothesis	F. — Confounding Variable	G. — Dynamic Analysis	H. — Formal Code Inspection
I. — Fuzz Testing	J. — Integration Testing	K. — Milestone	L. — Mock
M. — Pair Programming	N. — Passaround Code Review	O. — Perverse Incentives	P. — Race Condition
Q. — Regression Testing	R. — Risk	S. — Sampling Bias	T. — Software Metric
U. — Static Analysis	V. — Streetlight Effect	W. — Triage	X. — Unit Testing
Y. — Waterfall Model			

Q1.1: Johnnie is tasked with writing unit tests for a collection of classes in their company's codebase. Half of the classes are written in Python, and the other half is written in Go language. Johnnie is fluent in Python, but has never used Go. So, they spend a majority of their time making plenty of unit tests for the Python classes and thus find more bugs in those classes.

Q1.2: Tommie is the Chief Technology Officer at Bioplex, a music streaming service. They are currently working towards a complete overhaul of their mobile application's user interface. Tommie sets an internal deadline that 60% of the new screens should be implemented and tested by the end of the month. Although customers won't see this, it helps to keep the engineering team on track.

Q1.3: Domzoom wants to see if changing the color of the 'Buy Now' button on their checkout page to green will increase sales. So, they gave half the users the original red button, and the other half of users received the proposed green button.

Q1.4: Management at Kookle wants to release a video-editing software that is accessible and easy-to-use for beginners. After finishing the development of this product, Kamilla wants to test the usability of the software. So, Kamilla shares a working prototype with their friends who are experts in video-editing. The experts report the tool is very easy to use.

Q1.5: As an experienced product manager at Green-Plus, Robbie knows that there are bound to be issues that arise during the development of Green-Plus's newest flagship product. So, they list out the potential issues and mitigation strategies, so the product is released on time.

Q1.6: Jamie and Bill are partners for their EECS 280 project. They decide that, for each function, one of them will implement the code and the other will look for bugs and write test cases for that function. This style of programming makes them very effective at writing quality, bug-free code.

Q1.7: Condax, a social networking company, completes a working prototype for a new app to find people with similar interests as you. Before they release it to the public, they install the internal prototype on their engineers' phones to test and ensure the product works as expected.

Q1.8: Donware recently released a brand new app for their new product. In the first week since the launch of the new app, Donware's users reported over 100 bugs to the software engineering team. The lead software engineer prioritizes all the bugs, instructing their engineers to first fix high-risk bugs, such as security vulnerabilities.

Q1.9: Frankie wants to ensure their code will not have any 'divide by zero' errors, so they use a tool that scans their code for errors, without actually running the code itself.

Q1.10: A multithreaded program ensures that customers of Botsy, an online banking solution, can see their up-to-date bank balance. However, customers are reporting that their balances are incorrect. The engineers find that, since there are multiple threads updating the same variable without proper locking, the value of the balance may not always be accurate.

LATE

minutes remaining

Hide Time

Manual Save

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Extra Credit
- Pledge & Submit

Q1.11: All 8 software engineers at startup Opentech have a 5 hour long meeting to go through their newly implemented codebase and find bugs and discuss potential improvements.

Q1.12: Zencorp created a new product to allow musicians to edit their recorded music files. To ensure their software works as expected, they randomly generate 750 audio files (in different formats such as .mp3, .wav, and .m4a) for use in their testing to see if the editing features function correctly.

Q1.13: The startup that Jackie works for rewards its developers for the numbers of bugs they are able to fix. Jackie and their coworkers start to purposefully deploy buggy software so that they are later rewarded for fixing the bug.

Q1.14: An internal tool is created for software development leads at Plusstrip to view the maintainability index and cyclomatic complexity of all the classes that their engineers implemented. This aids them in quantifying the performance of their software.

Question 2. Code Coverage (20 points)

You are given the following C functions. Assume that statement coverage applies only to statements marked STMT_#. Importantly, in this question, we consider the entire program when calculating coverage. That is, even if the program execution starts from one particular method, we consider coverage with respect to the contents of all methods shown. Similarly, even if some methods are not executed during the program execution, we consider coverage with respect to the contents of all methods shown. Finally, we assume each input string (a and b) to aventura has at least one character; in other words, we do not consider NULL or empty strings.

```c
void aventura(str a, str b) {
    int pos = strlen(b) - 1;
    if (a[0] == b[pos]) {
        south_state(strlen(b), strlen(a));
        STMT_1;
    }
    STMT_2;
}

void south_state(int x, int y) {
    STMT_3;
    for (int i = 1; i < y; i++) {
        x += 1;
    }
    if (x < 8) {
        STMT_4;
        church_st(x, y);
    } else {
        STMT_5;
        x = x / 2;
        church_st(y, x);
    }
}

void church_st(int m, int n) {
    if (m  %  n == 0) {
        STMT_6;
        return;
    }
    else {
        church_st(m, m);
        STMT_7;
    }
    STMT_8;
}

```

(a) (3 points) In this question (including this sub-question and subsequent sub-questions), we define "one input" as one pair (a, b) of values. For example, ("this", "that") is one input where a is the string "this" and b is the string "that".
True / False: Suppose you can choose one input (a, b) to the function aventura. It is possible to achieve **95%** or greater *statement coverage*. Note that we assume that covering all of STMT_1 through STMT_8 counts as 100%.

LATE

minutes remaining

Hide Time

Manual Save

Navigation

○ True
○ False

(b(1)) (2 points) Provide **one** test input to `aventura` such that it achieves a *statement coverage* rate between **60%** and **96%** (inclusive).
Keep in mind that the statement coverage rate must be within the specified range. Write your test input in the form `("hello", "world")` followed by the corresponding coverage rate (separated by comma): for example, `("hello", "world"), 80%`. If you believe there is no such input that can achieve the target statement coverage rate, please explain. For your explanation, please limit your answer to at most four sentences.
In the context of this question, you have to pick inputs from the following five strings: `{ amers, avalon, isalita, kangs, mani }`.

Your answer here.

(b(2)) (3 points) Provide **another** test input to aventura such that it achieves a *statement coverage* rate between **60%** and **96%** (inclusive) and is **different** from the statement coverage in your answer for b(1). If your answer to b(1) is "no such input", you could skip this one, since the answer will also be "no such input". You do not need to explain again.
The requirement is the same as in b(1). That is, the statement coverage rate must be within the specified range. Write your test input in the form `("hello", "world")` followed by the corresponding coverage rate (separated by comma): for example, `("hello", "world"), 80%`. If you believe there is no such input that can achieve the target statement coverage rate, please explain. For your explanation, please limit your answer to at most four sentences.
In the context of this question, you have to pick inputs from the following five strings: `{ amers, avalon, isalita, kangs, mani }`.

Your answer here.

(c) (4 points) What is the **maximum** *branch coverage* achievable by exactly one input to `aventura`? Please briefly explain your answer.
Put the coverage rate in the first line (as a percentage or fraction), and then justification in the rest of lines. Limit your answer to at most four sentences.

Your answer here.

```
 1 void basketball(bool a, bool b) {
 2     if (a > b) { // line 2
 3         STMT_A;
 4     }
 5     if (a == b) { // line 5
 6         STMT_B;
 7     }
 8     STMT_C;
 9 }
10
```

(d) (4 points) Look at **line 5** in the `basketball` function above. Which operator should replace the `==` to achieve the **maximum** *path coverage* for a single input? In this case, we consider **four options:** `>`, `<`, `==`, `!=` when replacing the operator.
Please calculate the path coverage after doing the modification.
Note: In this question, we define "path coverage" as the proportion of routes through a given piece of code that have been executed, which is calculated among all *feasible* paths. A path is feasible if there exists an input that can exercise the path.
Please format your answer as two separate lines with your operator of choice followed by your calculated path coverage.

Your answer here.

(e) (4 points) Consider a company is attempting to implement some software to locate their bugs. They debate which coverage-based metric they should use as their mathematical basis.
Support or refute the claim that it would be better to use *branch coverage* counts rather than *statement coverage* counts.
Limit your answer to at most four sentences.

Your answer here.

LATE

minutes remaining

Hide Time

Manual Save

Question 3. Short Answer (3 points each, 15 points)

(a) (3 points)

Suppose you are interviewing at company 481Inc., and you get the following technical question:

You have a graph of n nodes labeled from 0 to n - 1. You are given an integer n and a list of edges where edges[i] = [a_i, b_i] indicates that there is an undirected edge between nodes a_i and b_i in the graph. Return true if the edges of the given graph make up a valid tree, and false otherwise.

What are TWO questions you can ask the interviewer before start implementing your solution? For each question, use at most two sentences.

Your answer here.

(b) (3 points)

Your friend Hotspur actively has been contributing to an open-source Github repository, and there are 5 levels of severity when it comes to bug reports (with 5 being the highest).

He filed a bug report to this repository, and it has been assigned with a severity level of 4. He assumed this bug would be addressed soon with this high severity level. Weeks later, when he realized this bug is still there, he comes to you and complains about the repository organizer`s inefficiency.

After taking EECS 481, you learned that a severe defect report may not be assigned with a high priority, and please list TWO reasons why this can happen. Please limit each reason to at most two sentences; your answer should use no more than four sentences.

Your answer here.

(c) (3 points)

Suppose you have a function which is known to be buggy. The reason you know it is buggy is because this function does not pass all the unit tests, which it is supposed to pass. This function is quite simple: it has only five statements. However, it is written in a really strange programming language that you have no idea about. As a result, it is extremely hard for you to inspect the program text manually. So, you cannot locate the buggy line manually by reading the code.

You decide to use Coverage-Based Fault Localization technique to automatically identify the buggy line(s). In particular, you plan to use the following suspiciousness score definition. (This definition is consistent with the suspiciousness ranking definition from the Fault Localization and Profiling lecture. But we typeset it using LaTeX here for more stable rendering.)

$$susp(s) = \frac{\frac{fails(s)}{total_{fail}}}{\frac{fails(s)}{total_{fail}} + \frac{pass(s)}{total_{pass}}}$$

You have collected the statements visited by each unit test below, together with the result (pass or fail) of each unit test.

- Test 1
  - Visits statements 1, 2, 3
  - Pass
- Test 2
  - Visits statements 1, 3, 4
  - Pass
- Test 3
  - Visits statements 1, 2, 5

LATE

minutes remaining

Hide Time

Manual Save

- Fail

Please list the suspiciousness score for each statement (rounded to the nearest hundredth).

For example:
Statement 1: 0.45,
Statement 2: 0,
Statement 3: 1,
Statement 4: 0.33,
Statement 5: 0.67

Your answer here.

(d) (3 points) From the guest lecture, Daniel Hoekwater shared a few motivations for developing your soft skills as a developer. Please list one of them. Limit your answer to at most four sentences.

Your answer here.

(e) (3 points)

Suppose you are working on a buggy multi-threaded C++ codebase. Your colleague suggests that both static analysis and dynamic analysis can be used to expose bugs.

Please list one example where static analysis can be used to expose a multi-threaded program AND one example where dynamic analysis can be used to expose a multi-threaded program. Limit your answer to at most four sentences.

Your answer here.

Question 4. Mutation Testing (13 points)

Consider the following Python implementation of insertion sort. There are two mutants introduced in the program and are shown in the comments below.
Particularly, mutant 1 decreases the upper bound of the for loop from n to n-1 and everything else stays the same.
Mutant 2 changes part of the 'while' conditional from using '<' to '>' and everything else stays the same.

```python
# Sorts the given list in ascending order
# This function returns the sorted list

def insertion_sort(arr):
    n = len(arr)
    if n <= 1:
        return

    for i in range(1, n):   # mutant 1: for i in range(1, n-1):
        key = arr[i]

        j = i-1
        while j >= 0 and key < arr[j]:  # mutant 2: while j >= 0 and key > arr[j]:
            arr[j+1] = arr[j]
            j -= 1
        arr[j+1] = key

    return arr

```

(a) (6 points)

You are given three test inputs. Please fill in the following table. Indicate in the table below whether or not each of these test inputs would kill each mutant.

Test #	Input	Oracle	Mutant 1 Killed?	Mutant 2 Killed?
1	arr = ['h', 'o', 'd', 'a', 'y']	['a', 'd', 'h', 'o', 'y']	○ True ○ False	○ True ○ False
2	arr = [3, 2, 1, 5, 4]	[1, 2, 3, 4, 5]	○ True ○ False	○ True ○ False
3	arr = [5, 5, 1, 5, 8]	[1, 5, 5, 5, 8]	○ True ○ False	○ True ○ False

LATE

minutes remaining

Hide Time

Manual Save

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Extra Credit
- Pledge & Submit

(b) (2 points) What is the mutation adequacy score (proportion not percentage) for a test suite containing tests 1 and 2 using mutants 1 and 2? Enter your answer as a decimal rounded to one decimal place (e.g. 0.1)

Your answer here.

(c) (2 points) What is the mutation adequacy score (proportion not percentage) for a test suite containing test 3 using mutants 1 and 2? Enter your answer as a decimal rounded to one decimal place (e.g. 0.1)

Your answer here.

(d) (3 points) Which test case(s) is/are the most/least useful or insightful. Briefly support your answer. Limit your answer to at most 4 sentences.

Your answer here.

Question 5. Invariants (8 points)

A totally anonymous 481 IA loves Pikachu, so they made a function to rate how awesome a given Pikachu picture is. Given some details about the image, the code calculates the Pikachu awesomenessScore. The code takes in 4 parameters and returns the awesomenessScore:

Variable	Explanation
is_Pikachu	True if the image contains a pikachu in it!
chonkiness	A positive number indicating the pikachu's chonkiness (a measure of its weight)
cuteness	A positive number indicating the pikachu's cuteness (a higher number indicates a cuter pikachu!)
is_Silly	True if the image contains something silly (pikachu giggles or acts crazy)

```
PikachuAwesomenessCalculator(bool is_Pikachu, double chonkiness, int cuteness, bool is_Silly):

    awesomenessRatio = 1 + chonkiness / cuteness; // floor division!
    awesomenessScore = 5

    while (is_Silly && chonkiness > 10){
        awesomenessScore = awesomenessScore + 1;
        chonkiness = chonkiness − 1;
        print('chonky')
    }

    // If pikachu is silly the picture is twice as awesome
    if (is_Silly){
        awesomenessRatio = awesomenessRatio ∗ 2
        print("silly_little_pika")
```

LATE

minutes remaining

Hide Time

Manual Save

Navigation

```
    } else {
        print('srs pika')
    }

    awesomenessScore = awesomenessScore * awesomenessRatio

    // Without pikachu it can not be awesome
    if (!is_Pikachu){
        awesomenessScore = 0
    }

  // Invariants Evaluated Here
  return awesomenessScore

}
```

(a) (8 points)

Consider the following four invariants generated by an oracle for `PikachuAwesomenessCalculator`. (The oracle here could be an automated dynamic invariant generation tool like Daikon. Or it could be created manually by a human. How this oracle is implemented is not important in this question. We just assume we're given four invariants.)

```
INV_1: !is_Pikachu || is_Silly
INV_2: !is_Pikachu || awesomenessScore >= 5
INV_3: awesomenessRatio > 1
INV_4: !(is_Pikachu && chonkiness > cuteness) || awesomenessScore >= 10
```

The invariants are evaluated at the end of the function – as indicated at the end of the `PikachuAwesomenessCalculator`. For each invariant, please first indicate either (1) the invariant is valid, or (2) the invariant is not valid.

Then, explain your reasoning. That is, for each invariant, if it is valid, please briefly explain why you believe it is valid. If invalid, please describe a situation in which this invariant is violated. For example, you could specify the parameter values that make the invariant invalid – please use the format like `[is_Pikachu = False, chonkiness = 0.1, cuteness = 2, is_Silly = True]`.

Please provide your answer for each invariant in the box below. Each invariant goes on a new line. You can use the format like `INV_1: valid-or-invalid. your-reasoning.` For each invariant, limit your answer to at most 1 sentence.

Your answer here.

Question 6. Dynamic Analysis (19 points)

(a) (4 points) Instrument the following Python code such that at the end of `main` it prints out the **total** time spent running each function (namely, `foo` and `baz`). Note that the total time running a function, say `fizz`, includes the time spent on its callees (that is, functions that are called by `fizz`) as well: for example, if `fizz` calls `buzz`, then time spent running `buzz` counts towards the time `fizz` was running.

You will use the following three helper functions for instruction:

- `resume_timer(timer_name)`: resume the timer with the name `timer_name`, or if the timer does not exist yet, start the timer. You do not need to worry about how to initialize the timer; the helper functions take care of it.
- `pause_timer(timer_name)`: pause the timer with the name `timer_name`.
- `print_timer(timer_name)`: print the current time recorded by the timer with the name `timer_name`.

As an example (note: this is just an example):

```
def fizz():
    x = 1
```

LATE

minutes remaining

Hide Time

Manual Save

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Extra Credit
- Pledge & Submit

```
    resume_timer(“hello_world”)
    print(“hello world!”)
    pause_timer(“hello_world”)
    y = x * x
    print(y)
    print_timer(“hello_world”)

```

The last print statement prints out the time spent on running the `print(“hello world!”)` statement.

Please modify the code below to instrument:

```
def foo(x, y):
    x = x + y
    x = x * y
    z = x + y
    baz(z)
    if y > x:
        return baz(x)
    return z

def baz(z):
    z = z + 7
    z = z * 3
    foo(z, 3)
    return z

def main():
    z = foo(1,3)
    s = baz(z)
    m = baz(s)
    print(“time spent on running foo:”)
    # add your print
    print(“time spent on running baz:”)
    # add your print

```

(b) (3 points) List and briefly explain 1 pro and 1 con of using static analysis instead of dynamic analysis to analyze the program's execution time. (Use 4 sentences or less for your answer.)

Your answer here.

Suppose that the previous program is part of a larger codebase, and suppose also that you have correctly instrumented the entirety of the codebase according to the instructions above (that is, you have added code that correctly logs function runtime to each function). Running your instrumentation on a file containing the functions `hi`, `hello`, and `goodbye` and assembling the data provides the following **cumulative** function runtime information (time is in seconds).

Cumulative time for **`hi`**	Cumulative time for **`hello`**	Cumulative time for **`goodbye`**
46	80	40

Based on the data from your outputted function log, select whether the following events could have occurred or not. Assume that functions are not recursive (do not call themselves) and assume also that each function was called at least once.

(c-i) (2 points) `goodbye` calls `hi`

○ True
○ False

(c-ii) (2 points) `hi` calls `goodbye` three times

LATE

minutes remaining

Hide Time

Manual Save

Navigation

○ True
○ False

(c-iii) (2 points) `hello` calls both `goodbye` and `hi` directly

○ True
○ False

(c-iv) (2 points) `goodbye` is called at least twice

○ True
○ False

(d-i) (2 points) Suppose you want to test what would happen to your data center infrastructure in case of a server failure, which tool would you use?

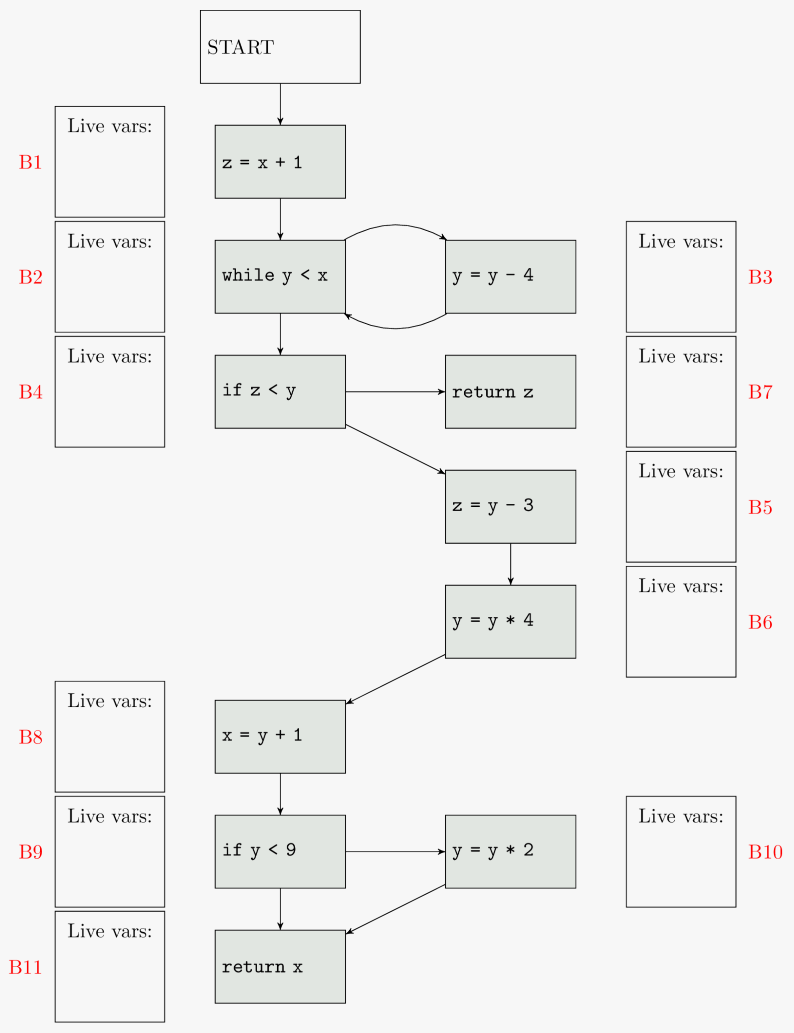
○ A) Chaos Monkey
○ B) Eraser
○ C) Chess
○ D) Driver Verifier

(d-ii) (2 points) Among the following scenarios, which one could admit an unsound analysis? In the context of this question, an unsound analysis is one that includes all positive results in its output but may also include some false results (i.e., false positives/alarms). In other words, if the goal is to find everything for which some property X holds, an unsound analysis will indeed find them all but it may also find things where X does not hold.

○ A) Creating medical software that detects the presence of a disease
○ B) Creating software that determines if a candidate is a good hire based on their interview performance
○ C) Creating software that determines if someone is a bad hire based on their interview performance
○ D) Creating software that predicts if a pricey business acquisition will pay off

Question 7: Dataflow Analysis (11 points total)

Consider a *live variable dataflow analysis* for three variables, `x`, `y`, and `z` used in the control-flow graph below. We associate with each variable a separate analysis fact: either the variable is (1) possibly read on a later path before it is overwritten (live), or (2) it is not (dead). We track the set of live variables at each point: for example, if `x` and `y` are alive but `z` is not, we write `{x, y}`. The special statement `return` reads, but does not write, its argument. In addition, `if` and `while` read, but do not write, all of the variables in their predicates. (You must determine if this is a forward or backward analysis.)



(1 point each) For each basic block `B1` through `B11`, write down the list of variables that are live *right before* the start of the corresponding block in the control flow graph above. Please list only the variable names in lowercase without commas or other spacing (e.g., use either `ab` or `ba` to indicate that `a` and `b` are alive before that block).

B1		B2		B3		B4	

B5	B6	B7	B8
B9	B10	B11	

LATE

minutes remaining

Hide Time

Manual Save

Navigation

- Question 1
- Question 2
- Question 3
- Question 4
- Question 5
- Question 6
- Question 7
- Extra Credit
- Pledge & Submit

Extra Credit

Each question below is for 1 point of extra credit unless noted otherwise. We are strict about giving points for these answers. No partial credit.

(1) What is your favorite part of the class so far?

Your answer here.

(2) What is your least favorite part of the class so far?

Your answer here.

(3) Which section are you officially enrolled in? What do you like about your section? What do you not like so much about? (2 points)

Your answer here.

(4) If you read any optional reading, identify it and demonstrate to us that you have read it critically. (2 points)

Your answer here.

(5) If you read any *other* optional reading, identify it and demonstrate to us that you have read it critically. (2 points)

Your answer here.

Honor Pledge and Exam Submission

You must check the boxes below before you can submit your exam.

☐ I have neither given nor received unauthorized aid on this exam.

☐ *I am ready to submit my exam.*

Note that your submission will be marked as late. You can still submit, and we will retain all submissions you make, but unless you have a documented extenuating circumstance, we will not consider this submission.

Submit My Exam

Once you submit, you will be able to leave the page without issue. Please don't try to mash the button.

The exam is graded out of 100 points.