LATE

minutes remaining

Hide Time

Manual Save

Navigation

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

Question 1. Word Bank Matching (1 point each, 16 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. — Adapter Design Pattern	B. — Bug Bounties	C. — Concurrency Bug	D. — Delegation
E. — Delta Debugging	F. — Factory Method Pattern	G. — Functional Requirement	H. — Mutation Testing
I. — Observer Design Pattern	J. — Perverse Incentive	K. — Profiling	L. — Quality Requirement
M. — Requirements Elicitation	N. — Risk Assessment	O. — Singleton Design Pattern	P. — Stakeholders
Q. — Strong Conflict	R. — Traceability	S. — Validation	T. — Verification
U. — Weak Conflict			

Q1.1: Robbie wants to ensure that their new business idea for Golddex is viable, so they discuss with people like customers, regulatory bodies, and the relevant leadership at Golddex

Q1.2: Carey wants to ensure their collection of test cases is strong enough to get full points on the Autograder. So, they purposely make small changes to the source code of their program to see if the test cases can detect there is a change.

Q1.3: After eliciting requirements from all the relevant stakeholders, Kamilla - the product manager at Sunnamplex - wants to ensure that the requirements are complete and accurately reflect the customers' needs. So, they conduct interviews and create prototypes to ensure their requirements are correct.

Q1.4: Jan creates a C++ class 'Animal', with subclasses 'Dog' and 'Cat'. They have a function called `MakeAnimal` that can construct 'Animal' objects (which are either 'Dog' or 'Cat'), though the actual subclass of the returned object is not revealed in the function signature of `MakeAnimal`.

Q1.5: Ollie is finishing an EECS project, but wants to *formally* ensure that their software is correct and satisfies the specification. Specifically, they want to prove the absence of a particular class of bugs. So, Ollie uses *this* technique (that is based on rigorous mathematical proofs) to prove correctness of their software.

Q1.6: Bioplex is releasing a new video game for mobile devices. As they do requirements elicitation, one stakeholder says 'players should not be able to see other users' match history', while another stakeholder says 'it should be possible for players to see other users' match history'.

Q1.7: Kan-code, a cloud-hosting solution for businesses, has the following goal - 'the platform should be secure and easy-to-use'.

Q1.8: Toughzap, a startup creating an AI-powered financial advisor, identifies potential issues they may face in the future, such as data breaches or incorrect predictions. For each issue, they develop a mitigation strategy.

Q1.9: Openlane is creating an eBook mobile application. One stakeholder provides the requirement that 'users' books should expire from their library after 60 days of first opening the book'. Another stakeholder, however, says 'books should not expire; users should be able to read any books they've purchased in the past'

Q1.10: Jessie notices an issue in their multi-threaded program where two threads are accessing the same variable, and one thread is writing. They use CHESS to detect this.

Q1.11: Statholdings wants their engineers to complete tasks quicker, so they give performance bonuses based on each employee's 'lines of code added' in hopes of getting them to code more. However, engineers start writing unreadable and duplicate code to boost their metric.

Q1.12: [ ] Domzoom is building an application that requires a global object used by multiple classes. They ensure that there is only one instance of this object across all the classes when the application is running.

Q1.13: [ ] Kris is working on a personal project that allows anyone that accesses their website to view the real-time price of a given stock. When a user enters a stock ticker, they are 'subscribed' to any changes for that stock. If the stock price changes, the new price is published to all users subscribed to it.

Q1.14: [ ] Donware is creating a tool for customers to track all their current shipping orders. In order to create a useful tool for their customers, they *gather input* from their customers and other stakeholders to better understand their needs and preferences.

Q1.15: [ ] Paula is creating a custom sorting function, which relies on a comparator function they wrote. The comparator function enables code reuse.

Q1.16: [ ] Frankie wants to find out the specific functions that are causing a longer-than-expected runtime in their application. So, they use *this technique* which tells them which functions take a long time to run.

LATE

minutes remaining

Hide Time

Manual Save

Navigation

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

Question 2. Delta Debugging (20 points)

Consider the scenarios below. For each scenario, specify if Delta Debugging can be used to solve the problem. If it can, select the answer choice marked "useful". If it cannot be used to solve the problem, select an assumption which may be violated (if multiple assumptions are violated, you can choose any one of them).

(a-i) (3 points) Suppose we have an online order form open for UMich undergraduate students with an offer for free t-shirts, but we want to make sure that we only count one submission per student. Each form includes a student's UMID. Using a list of these UMIDs, we hope to find the smallest subset that contains all duplicate UMIDs, such that we can remove the corresponding duplicate submissions. Interesting(x) returns true if the set contains a duplicate UMID.

- ○ A) Useful
- ○ B) Not monotonous
- ○ C) Not unambiguous
- ○ D) Not consistent

(a-ii) (3 points) We have a large bug database for a search engine that contains hundreds of bug reports. Each report consists of many lines of information, some of which may not be relevant to the bug. We want to simplify these bug reports by eliminating all lines that do not produce the bug, and keep only those that cause the bug to surface. Interesting(x) returns true on a subset of lines if including the line produces the bug.

- ○ A) Useful
- ○ B) Not monotonous
- ○ C) Not unambiguous
- ○ D) Not consistent

(a-iii) (3 points) We have a website consisting entirely of HTML files. The site currently crashes under certain conditions, and we want to know which subset of files cause the browser to crash. Interesting(x) returns true on the subset of files x if x causes a crash.

- ○ A) Useful
- ○ B) Not monotonous
- ○ C) Not unambiguous
- ○ D) Not consistent

Consider the following algorithm, Tribug Debugging, which functions in a very similar way as the Delta Debugging algorithm covered in the lecture and shares the same interface. That is, Tribug Debugging also takes two inputs that meet the following two preconditions: (1) P which is a list that is alone not interesting and (2) C which is a list such that the union of P and C is interesting.
It returns `smallest`, which represents a smallest subset of C such that the union of P and `smallest` is interesting. The key difference between Tribug Debugging and Delta Debugging is that, rather than splitting C in half as in Delta Debugging, Tribug Debugging partitions C into thirds. The following `tribug` function is an implementation of this algorithm.

```
def split(list):
    third = math.floor(len(list) / 3)
    // Note that C is a list, so order matters
    return list[:third], list[third:third*2], list[third*2:]

```

LATE

minutes remaining

Hide Time

Manual Save

Navigation

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

```
def simple_tribug(P, C):
    for element in C:
        if interesting(P + element): return element
    return C

def tribug(P, C):
    if len(C) < 3: return simple_tribug(P, C)

    C1, C2, C3 = split(C)
    if (interesting(P + C1)): return tribug(P, C1)
    if (interesting(P + C2)): return tribug(P, C2)
    if (interesting(P + C3)): return tribug(P, C3)

    smallest = union(
            tribug(P + C1, C2 + C3),
            tribug(P + C2, C1 + C3),
            tribug(P + C3, C1 + C2)
            )

    return smallest

```

Consider an interesting(x) function that returns True if x contains the elements 3 and 5. How many calls to `interesting` are made by `tribug` on each of the following inputs?

(b-i) (2 points) P = [], C = [3, 0, 5]

Your answer here.

(b-ii) (2 points) P = [], C = [5, 4, 3]

Your answer here.

(b-iii) (2 points) P = [], C = [5, 2, 8, 3]

Your answer here.

You have decided to test out the Tribug algorithm from 2b) by running it on an input and adding print statements to see the values of P, C, and which subset of C is interesting at each iteration. Adding your print statements has created the code in the box below.

```

def split(list):
    third = math.floor(len(list) / 3)
    return list[:third], list[third:third*2], list[third*2:]

def simple_tribug(P, C):
    for element in C:
        if interesting(P + element): return element
    return C

def tribug(P, C):
    print(“========= Top of Tribug =========”) // PRINT
    print(“P is:” + P + “ and C is: “ + C) // PRINT

    if len(C) < 3: return simple_tribug(P, C)

    C1, C2, C3 = split(C)

    if (interesting(P + C1)):
        print(“P U C1 is interesting”) // PRINT
        return tribug(P, C1)
    if (interesting(P + C2)):
        print(“P U C2 is interesting”) // PRINT
        return tribug(P, C2)
    if (interesting(P + C3)):
        print(“P U C3 is interesting”) // PRINT
```

LATE

minutes remaining

Hide Time

Manual Save

Navigation

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

```
        return tribug(P, C3)

    smallest = union(
            tribug(P + C1, C2 + C3),
            tribug(P + C2, C1 + C3),
            tribug(P + C3, C1 + C2)
            )

    return smallest

def main():
    result = tribug([], [7, 1, 3, 0])

    print("========= Main =========")
    print("result is: " + result )


```

Below is the resulting log from running the main method on the input [7, 1, 3, 0]. The `interesting(x)` function here returns true if x contains the element 3.

```
1
2  ================ Top of Tribug ================
3  P is [] and C is [7, 1, 3, 0]
4  ______2ci_______
5
6  ================ Top of Tribug ================
7  P is ___2cii___ and C is ___2cii___
8  ______2ciii_______
9
10 ================ Main ================
11 The result is: [3]
12
13
```

Given the code and the log above, answer the following questions. Note that some of the questions may not be specific to the code/log above and may test your understanding of the Delta Debugging algorithm in a general context.

(c-i) (1 points) Which of the followings should be filled in 2ci? (line 4 of the log output)

○ A) P U C1 is interesting
○ B) P U C2 is interesting
○ C) P U C3 is interesting
○ D) None (none of the above print statements are printed)

(c-ii) (1 points) Which of the followings should be filled in two blanks of 2cii? (line 7 of the log output)

○ A) [], [3, 0]
○ B) [3], [0]
○ C) [7, 1], [3, 0]

(c-iii) (1 points) Which of the followings should be filled in 2ciii? (line 8 of the log output)

○ A) P U C1 is interesting
○ B) P U C2 is interesting
○ C) P U C3 is interesting
○ D) None (none of the above print statements are printed)

(c-iv) (1 points) Unlike Delta Debugging, Tribug Debugging does not require that the Interesting(x) function be consistent on inputs x.

○ True
○ False

(c-v) (1 points) In the Delta Debugging algorithm as shown in the lecture, interference occurs when there is an error in the running of the Interesting(x) function, which interferes with the results.

○ True

○ False

LATE

minutes remaining

Hide Time

Manual Save

Navigation

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

Question 3. Short Answer (4 points each, 24 points)

(a) (4 points)

Your friend implemented a new automated program repair tool, GenMutantProg. In particular, GenMutantProg generates candidate higher-order mutants — from the original, buggy program using a new set of mutation operators designed by your friend — in hope that at least one of the candidate mutants will fix the bug from the original program and pass the test suite.

Explain what **"higher-order"** mutants mean in one sentence? Please also list **one advantage** and **one disadvantage** of generating higher-order mutants, compared to generating any mutants in general. Please use at most two sentences for the advantage and at most two sentences for the disadvantage.

Your answer here.

(b) (4 points)

Your team maintains a codebase and your manager proposes to apply readability metrics to each pull request with the goal of making sure the codebase contains highly readable code. Specifically, the plan is to calculate the readability score for the code in each pull request, and only those pull requests with a high readability score can be merged into the codebase.

Is this a good plan? Would this plan go wrong in any way? Please list **one possibility that this plan could go wrong**. Use at most two sentences in your answer.

Your answer here.

(c) (4 points)

Suppose you maintain a very large, currently closed-source project that is very popular among many users. At some point, you received too many feature requests and bug reports that you could not handle all by yourself alone. Now, you want to turn this project into an open-source one, with the goal of using contributors online to resolve issues (including implementing new features and/or fixing bugs).

However, one concern you have is that "random contributors" with different backgrounds may end up creating low-quality code (such as code that does not function properly, code that functions but has low readability, etc.). You don't want to allow online contributors to add such low-quality code into your project's codebase.

What is one approach that you can use to make sure no (or very little) low-quality code can be inserted into the codebase? Limit your answer to at most two sentences.

Your answer here.

(d) (4 points)

Imagine you are working on the autopilot feature for an airplane manufacturer. Once implemented, this feature will be deployed on all airplanes.

Please list **one informal quality requirement** and **one verifiable quality requirement** for this task.

Your answer here.

(e) (4 points)

LATE

minutes remaining

Hide Time

Manual Save

Navigation

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

WebRobot and FlashFill are both program synthesis techniques that generate programs from a domain-specific language (or DSL). **What is a DSL and how is it different from Python?** Use at most two sentences in your answer.

Also, how is WebRobot different from FlashFill in terms of the underlying synthesis approach? Please use at most two sentences for synthesis approach.

Your answer here.

(f) (4 points)

You have developed a tool, ErrorFixer481, that automatically fixes "Last-Mile Errors". In particular, ErrorFixer481 is based on a neural approach.

From the guest lecture presented by Jose Cambronero, what are Last-Mile Errors? What is one disadvantage of using a neural approach? Please limit your answer to 4 sentences.

Your answer here.

Question 4. Design Patterns (20 points)

For each code snippet below...
- Which design pattern does this code employ? [2 pts]
- What is one potential pro of employing this design pattern? Please limit your answer to one sentence. [1 pt]
- What is one potential con of employing this design pattern? Please limit your answer to one sentence. [1 pt]

(ai) (2 points)

```
void print_elements(Container<int>::traverse begin, Container<int>::traverse end) {
   for (Container<int>::traverse trav = begin; trav != end; ++trav) {
      cout << *trav << " ";
   }
   cout << endl;
}


int main() {
   ///VECTOR
   vector<int> my_vec {1, 2, 3, 4};
   print_elements(my_vec.begin(), my_vec.end());

   //MAP
   map<int,int> my_map;
   my_map.insert(pair<int, int>(1, 1));
   my_map.insert(pair<int, int>(2, 2));
   my_map.insert(pair<int, int>(3, 3));
   print_elements(my_map.begin(), my_map.end());

   //SET
   set<int> my_set;
   my_set.insert(1);
   my_set.insert(2);
   my_set.insert(3);
   print_elements(my_set.begin(), my_set.end());
}
```

- ○ A) Adapter Pattern
- ○ B) Composite Pattern
- ○ C) Proxy Pattern
- ○ D) Named Constructor Pattern
- ○ E) Factory Pattern
- ○ F) Singleton Pattern
- ○ G) Iterator Pattern

- ○ H) Observer Pattern
- ○ I) Template Pattern

LATE

minutes remaining

Hide Time

Manual Save

Navigation

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

(aii) (2 points) Pros/Cons

Your answer here.

(bi) (2 points)

```
struct HashMap {
    void Search() {}
    void Delete() {}
    void Insert() {}
    ...
    struct Vector {
        int size() {}
        void push_back() {}
        void pop_back() {}
        void resize() {}
        ...
        struct Array {
            ...
        };
    };
};

```

- ○ A) Adapter Pattern
- ○ B) Composite Pattern
- ○ C) Proxy Pattern
- ○ D) Named Constructor Pattern
- ○ E) Factory Pattern
- ○ F) Singleton Pattern
- ○ G) Iterator Pattern
- ○ H) Observer Pattern
- ○ I) Template Pattern

(bii) (2 points) Pros/Cons

Your answer here.

(c) (3 points) Briefly explain 2 creational design patterns from lecture in your own words and give a brief example of one being better than the other. Please limit your answer to a maximum of 4 sentences (ex. 2 per design pattern)

Your answer here.

(d) (3 points) Suppose that a particular software engineering project spends 40% of its lifetime effort on implementation, 45% of its lifetime effort on design and documentation, and 15% of its lifetime effort on testing. You are considering a design that would (a) decrease the effort required for implementation by 20% (for example, if implementation previously took 10 hours, but that effort is increased by 35%, it would now take 13.5 hours); however, adopting this design would also (b) increase the effort required for design and documentation by 10% and (c) increase the effort required for testing by 10%. Assume the project originally required 100 hours to complete over its lifetime and calculate the new hours required by the project with your new proposed design. Should the new design be adopted? Explain your reasoning with no more than 3 sentences.

Your answer here.

LATE
minutes remaining

Hide Time

Manual Save

Navigation

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

(e) (3 points) Identify one anti-pattern, and explain why it is an anti-pattern. Briefly identify a better pattern to use. Use at most 4 sentences.

Your answer here.

(f) (3 points) Suppose you and your partner are working on a large project together. Your partner encourages you not to add any inline comments, as they say it makes the code look messy. Do you think your project would benefit from inline comments? If you aren't able to have inline comments, what is another strategy you could use to make sure your code is accessible/maintainable for new team members? Briefly explain how this strategy would be helpful. Use at most 4 sentences.

Your answer here.

Question 5. Automated Program Repair (20 points)

Consider each of the following three scenarios in (a), (b), and (c). In each scenario, there is a subject program that fails some of the test cases from a given test suite. For each scenario, you can assume the test suite is very comprehensive and has very high quality, such that any program that passes this test suite is deemed correct.
Now you plan to use **Automated Program Repair (APR)** in each scenario to fix the bug(s) in each subject program. In particular, the APR tool you will use is based on mutation (which is similar to GenProg in the lecture). That is, the tool mutates the original subject program to generate mutants and checks if any of these generated mutants are correct by running each mutant on the test suite to see if it passes the test suite.
We will consider each of the three scenarios below.

(a) In the first scenario, you have a function that takes as input a string and returns another string. The desired behavior of a correct implementation is to return the reverse of the input string. For example, if the input string is "`eecs`", the desired output should be "`scee`". The test suite contains many tests that are used to check if the output is the reverse of the input. However, some of the tests are failing due to some bug(s) in the function. Above is the buggy `reserve_string` function.

```
def reverse_string(input_string):
    reversed_string = ""
    for i in range(0, len(input_string)):
        reversed_string += input_string[len(input_string) - i + 1]
    return reversed_string
```

(a-i) (2 points) Can you explain why it is wrong? Also show one input on which the program returns a wrong output.
You can choose any input you like. Use at most four sentences in your answer.

Your answer here.

(a-ii) (4 points) Recall that the APR tool is based on mutation. In particular, in each mutation step, the tool can only change an existing operator/constant to a different one. For example, it can change the "`+`" operator to "`-`", or it can change a constant "`0`" to "`1`", but it cannot do both in one single mutation step. Also suppose the tool can generate all first-order mutants, but cannot generate any higher-order mutants.
Can this APR tool fix the bug from the `reverse_string` program? Please justify your answer using at most four sentences.

Your answer here.

LATE

minutes remaining

Hide Time

Manual Save

Navigation

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

(b) The following class is based on the **Singleton Design Pattern**, where only one single instance of the ValueList class can be created. This ValueList instance stores a list of values and has functions that allow adding new values and fetching all existing values.
The test suite has many tests that check the functionality of all the functions in this class, but a number of them are failing now.

```
class ValueList:
    __instance = None

    @staticmethod
    def get_instance():
        if ValueList.__instance is None:
            return Singleton.__instance
        ValueList()

    def __init__(self):
        ValueList.__instance = self

    def add_value(self, value):
        self.values.append(value)

    def get_all_values(self):
        return self.values


```

(b-i) (2 points) Can you explain why the ValueList class implementation is wrong? A hint is that it has **no more than three** bugs. Please find as many as you can and explain them all. For each bug, please use no more than two sentences in your explanation.

Your answer here.

(b-ii) (1 points) Let's still consider the APR tool from the previous scenario (for the `reverse_string` program). Would that tool be able to fix the program in this scenario?

Your answer here.

(b-iii) (3 points) Suppose we have a slightly improved version of the APR tool. It is still based on mutation but in each mutation step, it can not only change one operator/constant to another, but also can add an operator/constant, add a new line of code, or delete an existing line. In other words, in one mutation step, it can generate more mutants than in the tool in the previous scenario. Note that it can still make only one single change in one single mutation step (but now this change can involve addition or deletion of code).
A second improvement is that, instead of only considering all first-order mutants as in the previous scenario, now it also generates mutants that need two mutation steps. For example, it can generate a mutant that requires first adding a new line and then changing another operator in the original program.
Can this more powerful APR tool fix the program? Please justify your answer using no more than four sentences.

Your answer here.

(c) In this third scenario, consider the following program.

LATE

minutes remaining

Hide Time

Manual Save

Navigation

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

```
int SuperCutePikachu(string color, int cuteness, bool is_electric) {
    return 0;
}
```

This program calculates the SuperPowerAbility score of a pikachu based on its color, cuteness and whether or not it is_electric. In particular, the calculation begins with 0 as the initial score. If the color is yellow, it adds 10; otherwise, it adds 5. Then it multiplies the current score by cuteness. Finally, if the pikachu is_electric, it adds 20.
For example, ("yellow", 25, true) would result in a SuperPowerAbility of (10*25)+20 = 270 Points! That's really powerful!
("blue",15,false) would result in a SuperPowerAbility of (5*15) = 75 Points. That's not very powerful.
In the program above, the developer created a placeholder for the implementation that returns the initial score 0, but they forgot to update this program. As a result, almost all test cases are currently failing.

(c-i) (2 points) In this scenario, we consider an even more powerful APR tool than the tools in two scenarios above. This new APR tool can generate higher-order mutants and it **does not limit** the number of mutation steps — this is the main difference from previous scenarios.. In each mutation step, it can change an existing operator/constant, change an existing line, add a new line, or delete an existing line (which is the same as in the second scenario). This new APR tool is clearly more powerful. Can this new tool repair the `SuperCutePikachu` program **in theory**? That is, mutate the program to one that can pass all tests in the test suite.

Your answer here.

(c-ii) (3 points) Please justify your answer in (c-i).
In particular, if your answer is "Yes", please explain why and list one potential issue that the APR tool might run into **in practice** ("in practice" here means if you actually deploy the tool and run it on the program to repair it).
If your answer is "No", please explain why and propose a potential extension to the APR tool such that it can **in theory** repair the program.
Limit your entire answer to at most 4 sentences.

Your answer here.

(c-iii) (3 points) In addition to using a program repair technique, what is another **automated** approach that you could potentially use to "repair" the `SuperCutePikachu` program above? Please briefly explain this approach and justify why it could work in this scenario. Also compare and contrast this approach with the automated program repair approach. Please use no more than 6 sentences in your response.

Your answer here.

Question 6. Extra Credit (1 point each)

(*Feedback*) What was your favorite topic or activity during the course?

(*Feedback*) What do you think we should do more of next semester (or what is the thing you would most recommend that we change for future semesters)?

LATE

minutes remaining

Hide Time

Manual Save

Navigation

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

(*Guest Lecture*) List one thing you learned from guest speaker José Cambronero of Microsoft that was not listed on an introductory summary slide or otherwise convince us that you paid careful attention during that lecture.

(*Guest Lecture*) List one thing you learned from guest speaker Roscoe Bartlett of Sandia National Laboratories that was not listed on an introductory summary slide or otherwise convince us that you paid careful attention during that lecture.

(*Optional Reading 1*) Identify a single optional reading that was assigned after Exam 1. Write two sentences about it that convince us you read it critically. (The most common student mistakes for these questions in Exam 1 were choosing a required reading instead of an optional reading or failing to "identify" or name the reading selected.)

(*Optional Reading 2*) Identify **another** optional reading that was assigned after Exam 1. Write two sentences about it that convince us you read it critically.

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.