**Workshop 4: Containers**

In this workshop, you will code three classes that are in composition and aggregation relations. The classes will simulate a very simplified form of managing reservations in a restaurant. The restaurant will manage a collection of reservations (composition); a messaging system will be used to send confirmations for the reservation.

**Learning Outcomes**

Upon successful completion of this workshop, you will have demonstrated the abilities to:

* design and code composition and aggregation class relationships
* use the member functions of the string class to parse a string into tokens based on simple rules
* design and code a class that manages a dynamically allocated array of pointers to objects

**Submission Policy**

The *in-lab* section is to be completed during your assigned lab section. It is to be completed and submitted by the end of the workshop period. If you attend the lab period and cannot complete the *in-lab* portion of the workshop during that period, ask your instructor for permission to complete the *in-lab* portion after the period. If you do not attend the workshop, you can submit the *in-lab* section along with your *at-home* section (see penalties below). **In order to get credit for the *in-lab* portion, you must be present in the lab for the entire duration of the lab.**

The *at-home* portion of the workshop is due on the day that is four days after your scheduled *in-lab* workshop (@ 23:59:59), **even if that day is a holiday**.

All your work (all the files you create or modify) must contain your name, Seneca email, student number and the date of completion (use the following template):

// Name:

// Seneca Student ID:

// Seneca email:

// Date of completion:

//

// I confirm that the content of this file is created by me,

// with the exception of the parts provided to me by my professor.

You are responsible to back up your work regularly.

**Late Submission Penalties**

The workshop can be submitted up to **1 (one) day** late (the day that is 5 days after the lab period); submissions received on this day are considered **late** and are subject to penalties:

* only *in-lab* portion submitted late (after the end of the lab period): 0 for *in-lab* portion, max 7/10 for the entire workshop.
* only *at-home* portion submitted late (more than 4 days after the lab period): max 4 for the *at-home* portion, max 7/10 for the entire workshop.
* both *in-lab* **and** *at-home* portions submitted late: max 4/10 for the entire workshop.
* when the submission closes, if the workshop is not complete, the mark for the entire workshop will be 0/10. The workshop is considered complete if there are two separate submissions (*in-lab* submission and *at-home* submission) containing the *in-lab code*, *at-home code* and *reflection*.

The submission is considered closed at the end of the day that is 5 (five) days after the lab period.

***In-Lab***

The in-lab portion of this workshop consists of modules:

* w4 (supplied)
* Reservation

Enclose all your source code within the sdds namespace and include the necessary guards in each header file.

**w4 Module (supplied)**

**Do not modify this module!** Look at the code and make sure you understand it.

**Reservation Module**

This module defines a class that holds information about a single reservation at a restaurant for a date/time in October.

Design and code a class named Reservation that should be able to store the following information (for each attribute, you can chose any type you think it's appropriate--you must be able to justify the decisions you have made):

* **reservation id**: an array of characters
* **the name on the reservation**
* **the email** that can be used to send the confirmation that the reservation can be honored or not
* **the number of people** in the party
* **the day** when the party is expected to come (for simplicity, the day is an integer between 1 and 31)
* **the hour** when the party is expected to come (for simplicity, the hour is an integer between 1 and 24)

***Public Members***

* a default constructor
* Reservation(const std::string& m\_res): A constructor that receives the reservation as a string; this constructor is responsible to extract the information about the reservation from the parameter and store it in the attributes of the instance. The parameter will always have the following format:

ID:NAME,EMAIL,PARTY SIZE,DAY,HOUR

This constructor should remove all spaces from the **beginning and end** of any token in the string.

When implementing the constructor, consider the following functions:

* [std::string::substr()](https://en.cppreference.com/w/cpp/string/basic_string/substr)
* [std::string::find()](https://en.cppreference.com/w/cpp/string/basic_string/find)
* [std::string::erase()](https://en.cppreference.com/w/cpp/string/basic_string/erase)
* [std::stoi()](https://en.cppreference.com/w/cpp/string/basic_string/stol)

***Friend Helpers***

* overload the insertion operator to insert the content of a reservation object into an **ostream** object:
  + if the hour is between 6AM and 9AM (inclusive), the kitchen serves breakfast:

Reservation ID: NAME <email> Breakfast on day DAY @ HOUR:00 for #PARTY\_SIZE people.

* if the hour is between 11AM and 3PM (inclusive), the kitchen serves lunch:

Reservation ID: NAME <email> Lunch on day DAY @ HOUR:00 for #PARTY\_SIZE people.

* if the hour is between 5PM and 9PM (inclusive), the kitchen serves dinner:

Reservation ID: NAME <email> Dinner on day DAY @ HOUR:00 for #PARTY\_SIZE people.

* at any other time the kitchen is closed and only drinks can be served:

Reservation ID: NAME <email> Drinks on day DAY @ HOUR:00 for #PARTY\_SIZE people.

* the name on the reservation should be printed on a field of size 10, aligned to the right
* the email on the reservation (including the characters < and >) should be printed on a field of size 20, aligned to the left.
* this operator should insert the endline character before exiting.

**Sample Output**

When the program is started with the command (the file data.txt is provided):

w4.exe data.txt

the output should look like:

Command Line:

--------------------------

1: w4.exe

2: data.txt

--------------------------

Reservations

--------------------------

Reservation RES-001: John <john@email.com> Drinks on day 3 @ 5:00 for 2 people.

Reservation RES-002: David <david@email.com> Breakfast on day 4 @ 6:00 for 1 people.

Reservation RES-003: Sara <sara@email.com> Breakfast on day 5 @ 7:00 for 2 people.

Reservation RES-004: Ana <ana@email.com> Breakfast on day 5 @ 8:00 for 1 people.

Reservation RES-005: John <john@email.com> Breakfast on day 4 @ 9:00 for 1 people.

Reservation RES-006: Vanessa <vanessa@email.com> Drinks on day 3 @ 10:00 for 2 people.

Reservation RES-007: Mike <mike@email.com> Lunch on day 4 @ 11:00 for 4 people.

Reservation RES-008: Mike <mike@email.com> Lunch on day 5 @ 12:00 for 8 people.

Reservation RES-009: Dan <dan@email.com> Lunch on day 3 @ 13:00 for 2 people.

Reservation RES-010: Donna <donna@email.com> Lunch on day 5 @ 14:00 for 5 people.

Reservation RES-011: Ana <ana@email.com> Lunch on day 4 @ 15:00 for 4 people.

Reservation RES-012: John <john@email.com> Drinks on day 5 @ 16:00 for 2 people.

Reservation RES-013: Sara <sara@email.com> Dinner on day 3 @ 17:00 for 6 people.

Reservation RES-014: Jennifer <jenn@email.com> Dinner on day 5 @ 18:00 for 6 people.

Reservation RES-015: Stan <stan@email.com> Dinner on day 4 @ 19:00 for 5 people.

Reservation RES-016: Chris <chris@email.com> Dinner on day 4 @ 20:00 for 3 people.

Reservation RES-017: Vanessa <vanessa@email.com> Dinner on day 4 @ 21:00 for 4 people.

Reservation RES-018: David <david@email.com> Drinks on day 5 @ 22:00 for 4 people.

Reservation RES-019: Chris <chris@email.com> Drinks on day 3 @ 23:00 for 1 people.

Reservation RES-020: Donna <donna@email.com> Drinks on day 4 @ 24:00 for 3 people.

--------------------------

**Submission (30%)**

To test and demonstrate execution of your program use the same data as shown in the output example above.

Upload your source code to your matrix account. Compile and run your code using the latest version of the g++ compiler (available at /usr/local/gcc/9.1.0/bin/g++) and make sure that everything works properly.

Then, run the following command from your account (replace profname.proflastname with your professor’s Seneca userid):

~profname.proflastname/submit 305XXX\_w4\_lab

and follow the instructions. Replace XXX with the section letter(s) specified by your instructor.

**⚠️Important:** Please note that a successful submission does not guarantee full credit for this workshop. If the professor is not satisfied with your implementation, your professor may ask you to resubmit. Resubmissions will attract a penalty.

***At-Home***

The *at-home* part of this workshop upgrades your *in-lab* solution to include two more modules:

* Restaurant
* ConfirmationSender

The module Reservation dosn't need any change.

**Restaurant Module**

Add a Restaurant module to your project. This module should maintain a dynamically allocated array of objects of type Reservation: Reservation\* (each element of the array is an object of type Reservation).

***Public Members***

* Restaurant(Reservation\* reservations[], size\_t cnt): a constructor that receives as a parameter an array of pointers to objects of type Reservation (i.e., each element of the array is a pointer). If you need a refresh on arrays of pointers, re-read the material from the last term (chapter **Abstract Base Classes**, section **Array of Pointers**).
  + this constructor should store copies of all reservations
* add any other special members that are necessary to manage a the resource
* size\_t size() const: return how many reservations are in the system.

***Friend Helpers***

* overload the insertion operator to insert the content of a Restaurant object into an **ostream** object:
  + if there are no reservations:

--------------------------

Fancy Restaurant

--------------------------

The object is empty!

--------------------------

* if there are reservations:

--------------------------

Fancy Restaurant

--------------------------

RESERVATION

RESERVATION

...

--------------------------

**ConfirmationSender Module**

Add a ConfirmationSender module to your project. The purpose of this module is to receive all the reservations from multiple restaurants, and contact the recipients with a confirmation message.

This module should maintain a dynamically allocated array of **pointers** to objects of type Reservation: const sdds::Reservation\*\* (each element of the array is a pointer to an object of type Reservation).

***Public Members***

* add any special members that are necessary to manage a the resource (the resource is an **array of pointers**; the array must be managed by this class, but the objects at the addresses stored in the array are managed outside the class)
* ConfirmationSender& operator+=(const Reservation& res): add the parameter to the array.
  + if res is already in the array, this operator does nothing
  + resize the array to make room for the res
  + store the **address** of res in the array (this function should not make copies of the parameter)
* ConfirmationSender& operator-=(const Reservation& res): remove the parameter from the array
  + if res is not in the array, this operator does nothing
  + search the array for res, set the pointer in the array to nullptr when res is found. **To challenge yourself, try to actually resize the array.**

***Friend Helpers***

* overload the insertion operator to insert the content of a ConfirmationSender object into an **ostream** object:
  + if there are no reservations to confirm:

--------------------------

Confirmations to Send

--------------------------

The object is empty!

--------------------------

* if there are reservations to confirm

--------------------------

Confirmations to Send

--------------------------

RESERVATION

RESERVATION

...

--------------------------

**Sample Output**

When the program is started with the command:

w4.exe data.txt

the output should look like:

Command Line:

--------------------------

1: w4.exe

2: data.txt

--------------------------

Reservations

--------------------------

Reservation RES-001: John <john@email.com> Drinks on day 3 @ 5:00 for 2 people.

Reservation RES-002: David <david@email.com> Breakfast on day 4 @ 6:00 for 1 people.

Reservation RES-003: Sara <sara@email.com> Breakfast on day 5 @ 7:00 for 2 people.

Reservation RES-004: Ana <ana@email.com> Breakfast on day 5 @ 8:00 for 1 people.

Reservation RES-005: John <john@email.com> Breakfast on day 4 @ 9:00 for 1 people.

Reservation RES-006: Vanessa <vanessa@email.com> Drinks on day 3 @ 10:00 for 2 people.

Reservation RES-007: Mike <mike@email.com> Lunch on day 4 @ 11:00 for 4 people.

Reservation RES-008: Mike <mike@email.com> Lunch on day 5 @ 12:00 for 8 people.

Reservation RES-009: Dan <dan@email.com> Lunch on day 3 @ 13:00 for 2 people.

Reservation RES-010: Donna <donna@email.com> Lunch on day 5 @ 14:00 for 5 people.

Reservation RES-011: Ana <ana@email.com> Lunch on day 4 @ 15:00 for 4 people.

Reservation RES-012: John <john@email.com> Drinks on day 5 @ 16:00 for 2 people.

Reservation RES-013: Sara <sara@email.com> Dinner on day 3 @ 17:00 for 6 people.

Reservation RES-014: Jennifer <jenn@email.com> Dinner on day 5 @ 18:00 for 6 people.

Reservation RES-015: Stan <stan@email.com> Dinner on day 4 @ 19:00 for 5 people.

Reservation RES-016: Chris <chris@email.com> Dinner on day 4 @ 20:00 for 3 people.

Reservation RES-017: Vanessa <vanessa@email.com> Dinner on day 4 @ 21:00 for 4 people.

Reservation RES-018: David <david@email.com> Drinks on day 5 @ 22:00 for 4 people.

Reservation RES-019: Chris <chris@email.com> Drinks on day 3 @ 23:00 for 1 people.

Reservation RES-020: Donna <donna@email.com> Drinks on day 4 @ 24:00 for 3 people.

--------------------------

R: Testing Constuctor

==========================

--------------------------

Fancy Restaurant

--------------------------

Reservation RES-001: John <john@email.com> Drinks on day 3 @ 5:00 for 2 people.

Reservation RES-002: David <david@email.com> Breakfast on day 4 @ 6:00 for 1 people.

Reservation RES-003: Sara <sara@email.com> Breakfast on day 5 @ 7:00 for 2 people.

Reservation RES-004: Ana <ana@email.com> Breakfast on day 5 @ 8:00 for 1 people.

Reservation RES-005: John <john@email.com> Breakfast on day 4 @ 9:00 for 1 people.

Reservation RES-006: Vanessa <vanessa@email.com> Drinks on day 3 @ 10:00 for 2 people.

Reservation RES-007: Mike <mike@email.com> Lunch on day 4 @ 11:00 for 4 people.

Reservation RES-008: Mike <mike@email.com> Lunch on day 5 @ 12:00 for 8 people.

Reservation RES-009: Dan <dan@email.com> Lunch on day 3 @ 13:00 for 2 people.

Reservation RES-010: Donna <donna@email.com> Lunch on day 5 @ 14:00 for 5 people.

Reservation RES-011: Ana <ana@email.com> Lunch on day 4 @ 15:00 for 4 people.

Reservation RES-012: John <john@email.com> Drinks on day 5 @ 16:00 for 2 people.

Reservation RES-013: Sara <sara@email.com> Dinner on day 3 @ 17:00 for 6 people.

Reservation RES-014: Jennifer <jenn@email.com> Dinner on day 5 @ 18:00 for 6 people.

Reservation RES-015: Stan <stan@email.com> Dinner on day 4 @ 19:00 for 5 people.

Reservation RES-016: Chris <chris@email.com> Dinner on day 4 @ 20:00 for 3 people.

Reservation RES-017: Vanessa <vanessa@email.com> Dinner on day 4 @ 21:00 for 4 people.

Reservation RES-018: David <david@email.com> Drinks on day 5 @ 22:00 for 4 people.

Reservation RES-019: Chris <chris@email.com> Drinks on day 3 @ 23:00 for 1 people.

Reservation RES-020: Donna <donna@email.com> Drinks on day 4 @ 24:00 for 3 people.

--------------------------

==========================

R: Testing Copy Constuctor

==========================

--------------------------

Fancy Restaurant

--------------------------

Reservation RES-001: John <john@email.com> Drinks on day 3 @ 5:00 for 2 people.

Reservation RES-002: David <david@email.com> Breakfast on day 4 @ 6:00 for 1 people.

Reservation RES-003: Sara <sara@email.com> Breakfast on day 5 @ 7:00 for 2 people.

Reservation RES-004: Ana <ana@email.com> Breakfast on day 5 @ 8:00 for 1 people.

Reservation RES-005: John <john@email.com> Breakfast on day 4 @ 9:00 for 1 people.

Reservation RES-006: Vanessa <vanessa@email.com> Drinks on day 3 @ 10:00 for 2 people.

Reservation RES-007: Mike <mike@email.com> Lunch on day 4 @ 11:00 for 4 people.

Reservation RES-008: Mike <mike@email.com> Lunch on day 5 @ 12:00 for 8 people.

Reservation RES-009: Dan <dan@email.com> Lunch on day 3 @ 13:00 for 2 people.

Reservation RES-010: Donna <donna@email.com> Lunch on day 5 @ 14:00 for 5 people.

Reservation RES-011: Ana <ana@email.com> Lunch on day 4 @ 15:00 for 4 people.

Reservation RES-012: John <john@email.com> Drinks on day 5 @ 16:00 for 2 people.

Reservation RES-013: Sara <sara@email.com> Dinner on day 3 @ 17:00 for 6 people.

Reservation RES-014: Jennifer <jenn@email.com> Dinner on day 5 @ 18:00 for 6 people.

Reservation RES-015: Stan <stan@email.com> Dinner on day 4 @ 19:00 for 5 people.

Reservation RES-016: Chris <chris@email.com> Dinner on day 4 @ 20:00 for 3 people.

Reservation RES-017: Vanessa <vanessa@email.com> Dinner on day 4 @ 21:00 for 4 people.

Reservation RES-018: David <david@email.com> Drinks on day 5 @ 22:00 for 4 people.

Reservation RES-019: Chris <chris@email.com> Drinks on day 3 @ 23:00 for 1 people.

Reservation RES-020: Donna <donna@email.com> Drinks on day 4 @ 24:00 for 3 people.

--------------------------

--------------------------

Fancy Restaurant

--------------------------

Reservation RES-001: John <john@email.com> Drinks on day 3 @ 5:00 for 2 people.

Reservation RES-002: David <david@email.com> Breakfast on day 4 @ 6:00 for 1 people.

Reservation RES-003: Sara <sara@email.com> Breakfast on day 5 @ 7:00 for 2 people.

Reservation RES-004: Ana <ana@email.com> Breakfast on day 5 @ 8:00 for 1 people.

Reservation RES-005: John <john@email.com> Breakfast on day 4 @ 9:00 for 1 people.

Reservation RES-006: Vanessa <vanessa@email.com> Drinks on day 3 @ 10:00 for 2 people.

Reservation RES-007: Mike <mike@email.com> Lunch on day 4 @ 11:00 for 4 people.

Reservation RES-008: Mike <mike@email.com> Lunch on day 5 @ 12:00 for 8 people.

Reservation RES-009: Dan <dan@email.com> Lunch on day 3 @ 13:00 for 2 people.

Reservation RES-010: Donna <donna@email.com> Lunch on day 5 @ 14:00 for 5 people.

Reservation RES-011: Ana <ana@email.com> Lunch on day 4 @ 15:00 for 4 people.

Reservation RES-012: John <john@email.com> Drinks on day 5 @ 16:00 for 2 people.

Reservation RES-013: Sara <sara@email.com> Dinner on day 3 @ 17:00 for 6 people.

Reservation RES-014: Jennifer <jenn@email.com> Dinner on day 5 @ 18:00 for 6 people.

Reservation RES-015: Stan <stan@email.com> Dinner on day 4 @ 19:00 for 5 people.

Reservation RES-016: Chris <chris@email.com> Dinner on day 4 @ 20:00 for 3 people.

Reservation RES-017: Vanessa <vanessa@email.com> Dinner on day 4 @ 21:00 for 4 people.

Reservation RES-018: David <david@email.com> Drinks on day 5 @ 22:00 for 4 people.

Reservation RES-019: Chris <chris@email.com> Drinks on day 3 @ 23:00 for 1 people.

Reservation RES-020: Donna <donna@email.com> Drinks on day 4 @ 24:00 for 3 people.

--------------------------

==========================

R: Testing Move Constuctor

==========================

--------------------------

Fancy Restaurant

--------------------------

The object is empty!

--------------------------

--------------------------

Fancy Restaurant

--------------------------

Reservation RES-001: John <john@email.com> Drinks on day 3 @ 5:00 for 2 people.

Reservation RES-002: David <david@email.com> Breakfast on day 4 @ 6:00 for 1 people.

Reservation RES-003: Sara <sara@email.com> Breakfast on day 5 @ 7:00 for 2 people.

Reservation RES-004: Ana <ana@email.com> Breakfast on day 5 @ 8:00 for 1 people.

Reservation RES-005: John <john@email.com> Breakfast on day 4 @ 9:00 for 1 people.

Reservation RES-006: Vanessa <vanessa@email.com> Drinks on day 3 @ 10:00 for 2 people.

Reservation RES-007: Mike <mike@email.com> Lunch on day 4 @ 11:00 for 4 people.

Reservation RES-008: Mike <mike@email.com> Lunch on day 5 @ 12:00 for 8 people.

Reservation RES-009: Dan <dan@email.com> Lunch on day 3 @ 13:00 for 2 people.

Reservation RES-010: Donna <donna@email.com> Lunch on day 5 @ 14:00 for 5 people.

Reservation RES-011: Ana <ana@email.com> Lunch on day 4 @ 15:00 for 4 people.

Reservation RES-012: John <john@email.com> Drinks on day 5 @ 16:00 for 2 people.

Reservation RES-013: Sara <sara@email.com> Dinner on day 3 @ 17:00 for 6 people.

Reservation RES-014: Jennifer <jenn@email.com> Dinner on day 5 @ 18:00 for 6 people.

Reservation RES-015: Stan <stan@email.com> Dinner on day 4 @ 19:00 for 5 people.

Reservation RES-016: Chris <chris@email.com> Dinner on day 4 @ 20:00 for 3 people.

Reservation RES-017: Vanessa <vanessa@email.com> Dinner on day 4 @ 21:00 for 4 people.

Reservation RES-018: David <david@email.com> Drinks on day 5 @ 22:00 for 4 people.

Reservation RES-019: Chris <chris@email.com> Drinks on day 3 @ 23:00 for 1 people.

Reservation RES-020: Donna <donna@email.com> Drinks on day 4 @ 24:00 for 3 people.

--------------------------

==========================

CS: Testing Constuctor

==========================

--------------------------

Confirmations to Send

--------------------------

The object is empty!

--------------------------

==========================

CS: Testing Operators

==========================

--------------------------

Confirmations to Send

--------------------------

Reservation RES-006: Vanessa <vanessa@email.com> Drinks on day 3 @ 10:00 for 2 people.

Reservation RES-017: Vanessa <vanessa@email.com> Dinner on day 4 @ 21:00 for 4 people.

Reservation RES-009: Dan <dan@email.com> Lunch on day 3 @ 13:00 for 2 people.

--------------------------

--------------------------

Confirmations to Send

--------------------------

Reservation RES-006: Vanessa <vanessa@email.com> Drinks on day 3 @ 10:00 for 2 people.

Reservation RES-017: Vanessa <vanessa@email.com> Dinner on day 4 @ 21:00 for 4 people.

--------------------------

==========================

CS: Testing Copy Constuctor

==========================

--------------------------

Confirmations to Send

--------------------------

Reservation RES-006: Vanessa <vanessa@email.com> Drinks on day 3 @ 10:00 for 2 people.

Reservation RES-017: Vanessa <vanessa@email.com> Dinner on day 4 @ 21:00 for 4 people.

--------------------------

--------------------------

Confirmations to Send

--------------------------

Reservation RES-006: Vanessa <vanessa@email.com> Drinks on day 3 @ 10:00 for 2 people.

Reservation RES-017: Vanessa <vanessa@email.com> Dinner on day 4 @ 21:00 for 4 people.

--------------------------

==========================

CS: Testing Move Constuctor

==========================

--------------------------

Confirmations to Send

--------------------------

The object is empty!

--------------------------

--------------------------

Confirmations to Send

--------------------------

Reservation RES-006: Vanessa <vanessa@email.com> Drinks on day 3 @ 10:00 for 2 people.

Reservation RES-017: Vanessa <vanessa@email.com> Dinner on day 4 @ 21:00 for 4 people.

--------------------------

==========================

**Reflection**

Study your final solution, reread the related parts of the course notes, and make sure that you have understood the concepts covered by this workshop. **This should take no less than 30 minutes of your time.**

Create a **text** file named reflect.txt that contains your detailed description of the topics that you have learned in completing this particular workshop and mention any issues that caused you difficulty and how you solved them. Include in your explanation—**but do not limit it to**—the following points:

* the difference between the implementations of a composition and an aggregation.
* the difference between the implementations of move and copy functions in a composition and an aggregation.

**Quiz Reflection**

Add a section to reflect.txt called **Quiz X Reflection**. Replace the **X** with the number of the last quiz that you received and list all questions that you answered incorrectly.

Then for each incorrectly answered question write your mistake and the correct answer to that question. If you have missed the last quiz, then write all the questions and their answers.

**Submission (30% for code, 40% for reflection)**

To test and demonstrate execution of your program use the same data as shown in the output example above.

Upload the source code and the reflection file to your matrix account. Compile and run your code using the latest version of the g++ compiler (available at /usr/local/gcc/9.1.0/bin/g++) and make sure that everything works properly.

Then, run the following command from your account (replace profname.proflastname with your professor’s Seneca userid):

~profname.proflastname/submit 305XXX\_w4\_home

and follow the instructions. Replace XXX with the section letter(s) specified by your instructor.

**⚠️Important:** Please note that a successful submission does not guarantee full credit for this workshop. If the professor is not satisfied with your implementation, your professor may ask you to resubmit. Resubmissions will attract a penalty.