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 reservation management for a restaurant. The restaurant will manage a collection of reservations (composition); a messaging system will 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 workshop is divided into two coding parts and one non-coding part:

- *Part 1*: worth 0% of the workshop's total mark, is optional and designed to assist you in completing the second part.
- *Part 2*: worth 100% of the workshop's total mark, is due on **Sunday at 23:59:59** of the week of your scheduled lab. Submissions of *Part 2* that do not contain the *reflection* are not considered valid submissions and are ignored.
- *reflection*: non-coding part, to be submitted together with *Part 2*. The reflection does not have marks associated to it, but can incur a**penalty of max 40% of the whole workshop's mark** if your professor deems it insufficient (you make your marks from the code, but you can lose some on the reflection).

Every file that you submit must contain (as a comment) at the top **your name**, **your Seneca email**, **Seneca Student ID** and the **date** when you completed the work.

If the file contains only your work, or work provided to you by your professor, add the following message as a comment at the top of the file:

> I have done all the coding by myself and only copied the code that my professor provided to complete my workshops and assignments.

If the file contains work that is not yours (you found it online or somebody provided it to you), **write exactly which part of the assignment are given to you as help, who gave it to you, or which source you received it from.** By doing this you will only lose the mark for the parts you got help for, and the person helping you will be clear of any wrong doing.

Compiling and Testing Your Program

All your code should be compiled using this command on `matrix`:

```
/usr/local/gcc/9.1.0/bin/g++ -Wall -std=c++17 -g -o ws file1.cpp file2.cpp ...
```

- `-Wall` : compiler will report all warnings
- `-std=c++17` : the code will be compiled using the C++17 standard
- `-g` : the executable file will contain debugging symbols, allowing *valgrind* to create better reports
- `-o ws` : the compiled application will be named `ws`

After compiling and testing your code, run your program as following to check for possible memory leaks (assuming your executable name is `ws`):

```
valgrind ws
```

To check the output, use a program that can compare text files. Search online for such a program for your platform, or use *diff* available on `matrix`.

If, when you try to compile/submit the workshop on matrix, you encounter an error regarding `/lib64/libstdc++.so.6: version 'CXXABI_1.3.9 not found'`, add the following line at the end of your `.bashrc` file (this is a hidden text file located in the home folder-- `~` --of your matrix account):

```
export LD_LIBRARY_PATH=/usr/local/gcc/9.1.0/lib64:$LD_LIBRARY_PATH
```

After you edit the file, logout and login again--this problem should go away.

Part 1 (0%)

The first 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 is appropriate--you must be able to justify the decisions you make):

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

Public Members - a default constructor - `Reservation(const std::string& res)` : A constructor that receives the reservation as a string; this constructor is responsible for extracting information about the reservation from the parameter and storing it in the instance's attributes. The parameter will always have the following format: `ID:NAME,EMAIL,PARTY_SIZE,DAY,HOUR` This constructor should remove all leading and trailing spaces from the **beginning and end** of any token in the string.

When implementing the constructor, consider the following functions: - std::string::substr() - std::string::find() - std::string::erase() - std::stoi()

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 returning control.

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.
-------------------------
```

Test Your Code

To test the 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 305_w4_p1
```

and follow the instructions.

This part represents a milestone in completing the workshop and is not marked!

Part 2 (100%)

The second part of this workshop upgrades your solution to include two more modules: - `Restaurant` - `ConfirmationSender`

The module `Reservation` doesn'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 the reservations stored
- `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 the resource (the resource is an **array of pointers**; your class must manage this array, but the objects at the addresses stored in the array are managed outside this class)
- `ConfirmationSender& operator+=(const Reservation& res)` : add the address of the reservation `res` to the array.
    - if the address of `res` is already in the array, this operator does nothing
    - resize the array to make room for `res`
    - store the **address** of `res` in the array (your function should not make copies of the reservation)
- `ConfirmationSender& operator-=(const Reservation& res)` : remove the address of the reservation `res` from the array
    - if the address of `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: ```cpp

Confirmations to Send

The object is empty!

`- if there are reservations to confirm` cpp

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 Constructor
==========================
```
```

```
--------------------------
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 Constructor
==========================
--------------------------
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-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 Constructor
==========================
--------------------------
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 Constructor
==========================
--------------------------
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 Constructor
==========================
--------------------------
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 Constructor
==========================
--------------------------
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 and the result is suggested to be at least 150 words in length.**

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.

To avoid deductions, refer to code in your solution as examples to support your explanations.

Submission

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 305_w4_p2
```

and follow the instructions.

:warning: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.