

Project: Simple Point Of Sale System

Latest Release

Milestones

Milestones due dates

Final project mark and due dates

Citation, Sources

See below for details:

For work that is done entirely by you (ONLY YOU)

For work that is done partially by you.

If you have helped someone with your code

Compiling and Testing Your Program

Adding alias to .bash\_profile on matrix

Project Implementation notes: Very Important, read carefully

The Point Of Sale system

Milestone 1

The PosApp Module

The features of the POS system

Implementation

The PosApp class

menu

# Project: Simple Point Of Sale System

## Latest Release

MS2

## Milestones

Milestone	Revision	Approximate Workload (days)	Overview	Comments
MS1	V0.9	5	<a href="#">Watch</a>	
MS2	V1.0	9		
MS3		10		
MS4		4		
MS5		14		

Your task for the project for this semester is to create simple Point of Sale (POS) application that keeps track of a small inventory of Goods to sell and can sell them at the cashier, issuing a bill of sale.

## Milestones due dates

This project will be done in 5 milestones and each milestone will have its due date. The due date of each milestone is stated below, and it is based on the amount of work to be done for that milestone.

## Final project mark and due dates

Milestone	Mark	Due date	Submission Policy
MS1	10%	Mar 10	gets full mark even if 1 week late. gets 0% afterwards
MS2	10%	Mar 20	gets full mark even if 1 week late. gets 0% afterwards

Project: Simple Point Of Sale System

Latest Release

Milestones

Milestones due dates

Final project mark and due dates

Citation, Sources

See below for details:

For work that is done entirely by you (ONLY YOU)

For work that is done partially by you.

If you have helped someone with your code

Compiling and Testing Your Program

Adding alias to .bash\_profile on matrix

Project Implementation notes: *Very Important, read carefully*

The Point Of Sale system

Milestone 1

The PosApp Module

The features of the POS system

Implementation

The PosApp class


menu

Milestone	Mark	Due date	Submission Policy
MS3	10%	Mar 29	gets full mark even if 1 week late. gets 0% afterwards
MS4	10%	Apr 2	gets full mark even if 1 week late. gets 0% afterwards
MS5	60%	Apr 16	See below

To make the final submission of the project easier and to make it possible to partially submit a project we have divided the submission of milestone 5 into five small ones. Each submission is worth 12% of the project mark. Your project will be marked only if you have all four milestones and at least have one of the five submissions of milestone 5.

Milestone 5 Divided into five submission	Mark	Due date	Submission Policy
m51	12%	Apr 16th	10% penalty for each day being late up to 5 days
m52	12%	Apr 16th	10% penalty for each day being late up to 5 days
m53	12%	Apr 16th	10% penalty for each day being late up to 5 days
m54	12%	Apr 16th	10% penalty for each day being late up to 5 days
m55	12%	Apr 16th	10% penalty for each day being late up to 5 days

The first 4 milestones will not be marked based on the code, but their success and their timely submissions. You may modify or debug your previous code as you are going through the milestones. The only milestone that is going to scrutinized based your code will be milestone 5. If you require any feedback on your first four milestones you need to ask your professor to do so.

 The first four milestones must be submitted successfully even if they are very late. Your project will receive a mark of zero if any of the first 4 milestones are not submitted by the rejection date (Apr 21) . For your project to be marked, you must submit all the 4 milestones and at least one of the 5 submissions of Milestone 5 (Rejection date for milestone 5 is also Apr 21)

You can check the due date of each milestone using the `-due` flag in the submission command:

`~profname.proflastname/submit 2??/prj/m? -due`

## Project: Simple Point Of Sale System

### Latest Release

### Milestones

### Milestones due dates

### Final project mark and due dates

### Citation, Sources

See below for details:

For work that is done entirely by you (ONLY YOU)

For work that is done partially by you.

If you have helped someone with your code

### Compiling and Testing Your Program

Adding alias to

`.bash_profile` on matrix

Project Implementation notes:  
*Very Important, read carefully*

### The Point Of Sale system

### Milestone 1

### The PosApp Module

The features of the POS system

Implementation

The PosApp class

menu

- replace **2??** with the subject code
- replace **m?** with the milestone number

Different sections may have different due dates based on their section's progress. Always check the due date using this command. The due date returned by this command always have priority over the due dates stated above.

## Citation, Sources

When submitting your work, all the files submitted should carry full student information along with the "citation and sources" information. See the following example:

If you have multiple submissions of the same milestone, please update the Revision History in each submission so your professor knows what changes to look for.

```
/* Citation and Sources...
```

```
Final Project Milestone ?
```

```
Module: Whatever
```

```
Filename: Whatever.cpp
```

```
Version 1.0
```

```
Author John Doe
```

```
Revision History
```

```
-----
Date      Reason
2020/?/?  Preliminary release
2020/?/?  Debugged DMA
-----
```

I have done all the coding by myself and only copied the code that my professor provided to complete my project milestones.

```
-----
OR
-----
```

Write exactly which part of the code is given to you as help and who gave it to you, or from what source you acquired it.

```
-----*/
```

## Project: Simple Point Of Sale System

### Latest Release

### Milestones

### Milestones due dates

### Final project mark and due dates

### Citation, Sources

See below for details:

For work that is done entirely by you (ONLY YOU)

For work that is done partially by you.

If you have helped someone with your code

### Compiling and Testing Your Program

Adding alias to

`.bash_profile` on matrix

Project Implementation notes:  
*Very Important, read carefully*

### The Point Of Sale system

### Milestone 1

### The PosApp Module

The features of the POS system

Implementation

The PosApp class

menu

Failing to include the above citation to any of the files containing your work will cause the rejection of your project submission

See below for details:

### For work that is done entirely by you (ONLY YOU)

If the file contains only your work or the 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 project milestones.
```

### For work that is done partially by you.

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 is 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 wrongdoing.



This [Submission Policy](#) only applies to the project. All other assessments in this subject have their own submission policies.

### If you have helped someone with your code

If you have helped someone with your code. Let them know of these regulations and in an email write exactly which part of your code was copied and who was the recipient of this code.

By doing this you will be clear of any wrongdoing if the recipient of the code does not honour these regulations.

## Compiling and Testing Your Program

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

```
g++ -Wall -std=c++11 -g -o ms file1.cpp file2.cpp ...
```

- `-Wall` : the compiler will report all warnings
- `-std=c++11` : the code will be compiled using the C++11 standard

## Project: Simple Point Of Sale System

### Latest Release

### Milestones

### Milestones due dates

### Final project mark and due dates

### Citation, Sources

See below for details:

For work that is done entirely by you (ONLY YOU)

For work that is done partially by you.

If you have helped someone with your code

### Compiling and Testing Your Program

Adding alias to  
.`bash_profile` on matrix

Project Implementation notes:  
*Very Important, read carefully*

### The Point Of Sale system

### Milestone 1

### The PosApp Module

The features of the POS system

### Implementation

The PosApp class

menu

- `-g` : the executable file will contain debugging symbols, allowing *valgrind* to create better reports
- `-o ms` : the compiled application will be named `ms`

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

```
valgrind --show-error-list=yes --leak-check=full --show-leak-kinds=all --track-origin
```

- `--show-error-list=yes` : show the list of detected errors
- `--leak-check=full` : check for all types of memory problems
- `--show-leak-kinds=all` : show all types of memory leaks identified (enabled by the previous flag)
- `--track-origins=yes` : tracks the origin of uninitialized values ( `g++` must use `-g` flag for compilation, so the information displayed here is meaningful).

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`.

### Adding alias to `.bash_profile` on matrix

You can add the following two lines to the end of your `.bash_profile` file in your matrix home directory to create two aliases for the above C++ compilation and *valgrind* testing so you don't have to type such a long command to compile and execute your programs:

```
alias msc++="g++ -Wall -std=c++11 -g -o ms"
alias vrun="valgrind --show-error-list=yes --leak-check=full --show-leak-kinds=all --
```

By adding the above to the '`.bash_profile`' (you must logoff and log back it so it takes effect) you can compile your c++ files as follows:

```
msc++ file1.cpp file2.cpp file3.cpp...<ENTER>
```

The above will compile your files and if successful it will create an executable called `ms`.

```
vrun ms<ENTER>
```

## Project: Simple Point Of Sale System

### Latest Release

### Milestones

### Milestones due dates

### Final project mark and due dates

### Citation, Sources

See below for details:

For work that is done entirely by you (ONLY YOU)

For work that is done partially by you.

If you have helped someone with your code

### Compiling and Testing Your Program

Adding alias to  
[.bash\\_profile](#) on matrix

Project Implementation notes:  
*Very Important, read carefully*

### The Point Of Sale system

### Milestone 1

### The PosApp Module

The features of the POS system

### Implementation

The PosApp class

menu

The above will execute your `ms` executable under the tester `valgrind` for detailed test on leaks and runtime errors.

Make sure that all the debugging code and debugging comments are removed before submission.

## Project Implementation notes: *Very Important, read carefully*

- All the code written in this project should be within the namespace `sdds`.
- You are free and encouraged to add any attributes(member variables), functions and methods (member functions) you find necessary to complete your code. If you are not sure about your strategy for adding functionalities and properties to your classes, ask your professor for advice.
- If any methods are being added and they are not called outside the scope of the class, make sure they are private.
- Unless you are asked for a specific definition, name the variables, and functions yourself. Use proper names and follow the naming conventions instructed by your professor. Having meaningless and misleading names will attract a penalty.
- When creating methods (member functions) make sure to make them constant if in their logic, they are not modifying their class.
- If an Empty state is required for an object, it is considered to be an "invalid" empty state, and objects in this state should be rendered unusable.
- A module called **Utils** is added to the project that can be used for your custom functions and classes in your implementation. Leave this module empty if you don't have any custom functionalities. You can add any custom code/class of your own to the **Utils** module to be used throughout the project.
- Throughout the project, if any class is capable of printing, displaying or writing itself, the member function will always have the following signature:  
 The function will return a reference of an **ostream** and will receive a reference on an **ostream** as an optional argument. If this argument is not provided, the object **"cout"** will be passed instead.
- Throughout the project, if any class is capable of reading or receiving its content from a stream, the member function will always have the following signature:  
 The function will return a reference of an **istream** and will receive a reference on an **istream** as an optional argument. If this argument is not provided, the object **"cin"** will be passed instead.

## Project: Simple Point Of Sale System

### Latest Release

### Milestones

### Milestones due dates

### Final project mark and due dates

### Citation, Sources

See below for details:

For work that is done entirely by you (ONLY YOU)

For work that is done partially by you.

If you have helped someone with your code

### Compiling and Testing Your Program

Adding alias to

`.bash_profile` on matrix

Project Implementation notes:  
*Very Important, read carefully*

## The Point Of Sale system

### Milestone 1

#### The PosApp Module

The features of the POS system

#### Implementation

The PosApp class

menu

- When passing an object or variable by address or reference, if they are not to be modified, make sure they are passed as constant pointers and references.
- You may reuse and copy any code your professor provided for your workshops or functions you may have from previous work in this subject or other subjects and place it in the Utils module.

## The Point Of Sale system

We will start the development of the project by creating its user interface (MS1, PosApp module). Through this, we will create a mockup application that represents what the application will look like and act without actually doing anything. In short, we are creating a prototype of the application (in MS1) then coding its engine (in MS2, MS3 and MS4) and finally adding functionality to the POS system (MS1) to make it fully functional (in ms5, parts 1 to 5).

## Milestone 1

### The PosApp Module

This Module contains one Module called `PosApp`. This module is responsible to offer the user a menu of features provided by the POS system.

The user, then, can select an option representing the feature and execute it. After the execution is complete, the system displays the menu again, until the user selects to exit the application.

### The features of the POS system

1. List Items
2. Add Item
3. Remove Item
4. Stock Item
5. Point Of Sale

### Implementation

For milestone 1 you are responsible to create a mockup module for the Point Of Sale that will demonstrate how the system is going to run (eventually) by only printing the name of the actions, instead of executing them. In later



## Project: Simple Point Of Sale System

### Latest Release

#### Milestones

#### Milestones due dates

#### Final project mark and due dates

#### Citation, Sources

See below for details:

For work that is done entirely by you (ONLY YOU)

For work that is done partially by you.

If you have helped someone with your code

#### Compiling and Testing Your Program

Adding alias to

`.bash_profile` on matrix

#### Project Implementation notes:

*Very Important, read carefully*

### The Point Of Sale system

#### Milestone 1

#### The PosApp Module

The features of the POS system

#### Implementation

#### The PosApp class

##### menu

stages of development, you will replace these messages with the proper logic to actually perform the action.

Note that the signature of many of the methods created now will be changed to accommodate what needs to be done.

### The PosApp class

In module `PosApp` create a class with the same name ( `PosApp` ) having the following mandatory methods:

#### menu

This method will display the menu of the system, and receive the user's choice (in a foolproof way and return the choice). See below:

The Sene-Store

1- List items

2- Add item

3- Remove item

4- Stock item

5- Point of Sale

0- exit program

> a

Invalid Integer, try again: -1

[0<=value<=5], retry: > 6

[0<=value<=5], retry: > 1

Running `listItems()`

#### Mockup methods:

Create the following 7 methods that only print `Running` and their names.

1. `addItem()`

2. `removeItem()`

3. `stockItem()`

4. `listItems()`

5. `POS()`

6. `saveRecs()`



## Project: Simple Point Of Sale System

### Latest Release

#### Milestones

#### Milestones due dates

#### Final project mark and due dates

#### Citation, Sources

See below for details:

For work that is done entirely by you (ONLY YOU)

For work that is done partially by you.

If you have helped someone with your code

#### Compiling and Testing Your Program

Adding alias to

`.bash_profile` on matrix

Project Implementation notes:

*Very Important, read carefully*

#### The Point Of Sale system

#### Milestone 1

#### The PosApp Module

The features of the POS system

#### Implementation

The PosApp class

menu

#### 7. loadRecs()

#### Construction

PosApp is created by receiving the name of a file that is stored in character cString with a maximum length of 255 characters.

PosApp Can neither get copied nor assigned to another PosApp object. Your code must prevent such actions.

#### run method

implement the following actions calling the corresponding mockup methods

This method first loads all the records and then displays the menu waiting for the user to make the selection. After the (foolproof) selection the proper action is executed and again the menu is displayed until the option exit is selected. In the latter case, all the records are saved and a Goodbye! message is displayed.

See the [correct\\_output.txt](#) file for sample execution.

The tester file for this output is [main.cpp](#)

## MS1 Submission

#### files to submit

PosApp.cpp

PosApp.h

Utils.cpp

Utils.h

main.cpp

#### Data entry

abc

-1

6

1

2

## Project: Simple Point Of Sale System

### Latest Release

#### Milestones

#### Milestones due dates

#### Final project mark and due dates

#### Citation, Sources

See below for details:

For work that is done entirely by you (ONLY YOU)

For work that is done partially by you.

If you have helped someone with your code

#### Compiling and Testing Your Program

Adding alias to  
`.bash_profile` on matrix

Project Implementation notes:  
*Very Important, read carefully*

### The Point Of Sale system

#### Milestone 1

#### The PosApp Module

The features of the POS system

#### Implementation

The PosApp class

menu

3  
4  
5  
0

## Submission Process

If you would like to successfully complete the project and be on time, **start early** and try to meet all the due dates of the milestones.

Upload your source code and the tester program to your `matrix` account. Compile and run your code using the `g++` compiler [as shown in the introduction](#) 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 2??/prj/m1
```

and follow the instructions.

- 2?? is replaced with your subject code

## The submit program's options:

```
~ prof_name.prof_lastname/submit DeliverableName [-submission options]<ENTER>
[-submission option] acceptable values:
```

**"-due":**

Shows due dates only

This option cannot be used **in** combination with any other option.

**"-skip\_spaces":**

Do the submission regardless of incorrect horizontal spacing.

This option may attract penalty.

**"-skip\_blank\_lines":**

Do the submission regardless of incorrect vertical spacing.

This option may attract penalty.

## Project: Simple Point Of Sale System

### Latest Release

#### Milestones

#### Milestones due dates

#### Final project mark and due dates

#### Citation, Sources

See below for details:

For work that is done entirely by you (ONLY YOU)

For work that is done partially by you.

If you have helped someone with your code

#### Compiling and Testing Your Program

Adding alias to  
[.bash\\_profile](#) on matrix

Project Implementation notes:  
*Very Important, read carefully*

## The Point Of Sale system

### Milestone 1

#### The PosApp Module

The features of the POS system

#### Implementation

#### The PosApp class

#### menu

**"-feedback":**

Check the program execution without submission.

## Back to milestones

# Milestone 2

Start the second milestone by creating a general header file for the constant values needed throughout the development of the project.

The name of this file is `POS.h`

Add the following constant values:

`TAX: 0.13`

`MAX_SKU_LEN: 7`

`MIN_YEAR: 2000`

`MAX_YEAR: 2030`

`MAX_STOCK_NUMBER: 99`

`MAX_NO_ITEMS: 2000`

We may add more values to this header file later. If you need any constant value for your code, add it here.

## The Error Class

Create a class called Error. This class will be used as a smart flag in our future classes to keep track of the erroneous state of the object and the error message associated with it.

The Error class has only a character pointer as an attribute that holds the description of the error (the error message) in a dynamic C-string. (We will call this pointer the **error message pointer** from now on).

Note: naturally if the error message pointer is null, then the object is in a **No Error** state and if the error message is not null, then there is an error and the description of the Error is pointed by the error message.

## Project: Simple Point Of Sale System

### Latest Release

#### Milestones

#### Milestones due dates

#### Final project mark and due dates

#### Citation, Sources

See below for details:

For work that is done entirely by you (ONLY YOU)

For work that is done partially by you.

If you have helped someone with your code

#### Compiling and Testing Your Program

Adding alias to

`.bash_profile` on matrix

Project Implementation notes:

*Very Important, read carefully*

#### The Point Of Sale system

#### Milestone 1

#### The PosApp Module

The features of the POS system

#### Implementation

#### The PosApp class

#### menu

## Error Construction

- The Error class by default is in a "No Error" state. (its pointer is null)
- The Error class can also get created using an error message (a C-string). In this case, the Error object will be in an erroneous state.

## Rule of three

The Error class can be copied and assigned to another Error object safely and must get destructed with no memory leak.

## Mandatory Operations

### Assignment to a C-string.

An Error object should be able to be assigned to a C-string. This will set the error message to the content of the C-string dynamically or nullptr if the C-String is null or empty.

### boolean type conversion

If an Error object is casted to a **bool**, it should return true if the message is not null and false if it is null.

```
Error err;
...
...
if(err){
    // there is an error
}
else{
    // there is no error
}
```

### clear()

This method clears the error message and returns the reference of the current Error object. After this, the Error object should be in a "no error" state.

### ostream insertion

## Project: Simple Point Of Sale System

### Latest Release

### Milestones

### Milestones due dates

### Final project mark and due dates

### Citation, Sources

See below for details:

For work that is done entirely by you (ONLY YOU)

For work that is done partially by you.

If you have helped someone with your code

### Compiling and Testing Your Program

Adding alias to  
[.bash\\_profile](#) on matrix

Project Implementation notes:  
*Very Important, read carefully*

### The Point Of Sale system

### Milestone 1

### The PosApp Module

The features of the POS system

### Implementation

### The PosApp class

### menu

... ..

The Error message should be able to be inserted into an ostream object using the **operator<<**. If the Error object is not in an error state nothing will be inserted into the ostream object.

DO NOT use *friend* to implement this.

## Other Methods and operation

Implement and add any other method or operations you find necessary to accomplish the above tasks (make them private if they don't need to be publicly accessible)

## the Date class

### attributes

- The date class encapsulates date and time properties in five integer attributes (year, month, day, hour and minute).
- Date also has a flag for using date only (no hour or minutes)
- Finally, Date has an Error attribute to keep track of the success of operations.

### Date validation process

During the implementation of the Date class whenever validation is needed the attributes of the date are all checked in the following order and the Error object is set to the corresponding error message stated below.

### Date values

The valid range of values for the year is defined in `POS.h`. Month values are between 1 and 12 and day values are between 1 and the return value of the `daysOfMonth` function

### Time values

The valid range for time value is 0 to 23 for the hour and 0 to 59 for the minute.

### Validation Sequence

Note that as soon as a validation fails, the Error object is set, and the rest of the validations will not take place

1. If the year is invalid, then the Error object is set to "Invalid Year"
2. If the month is invalid, then the Error object is set to "Invalid Month"

## Project: Simple Point Of Sale System

### Latest Release

#### Milestones

#### Milestones due dates

#### Final project mark and due dates

#### Citation, Sources

See below for details:

For work that is done entirely by you (ONLY YOU)

For work that is done partially by you.

If you have helped someone with your code

#### Compiling and Testing Your Program

Adding alias to  
[.bash\\_profile](#) on matrix

Project Implementation notes:  
*Very Important, read carefully*

#### The Point Of Sale system

#### Milestone 1

#### The PosApp Module

The features of the POS system

#### Implementation

#### The PosApp class

#### menu

... ..

3. If the day is invalid, then the Error object is set to "Invalid Day"
4. If the hour is invalid, then the Error object is set to "Invalid Hour"
5. If the minute is invalid, then the Error object is set to "Invalid Minute"

## Already implemented utility functions

The following functions are already implemented to help you with implementing different functionalities in the Date class. Use them (or their logic) wherever you find fit or necessary.

### system date and time

The following function retrieves the current date and time of the system and returns them through the argument list.

```
#define _CRT_SECURE_NO_WARNINGS
#include <ctime>

void getSystemDate(int& year, int& mon, int& day, int& hour, int& min, bool dateOn)
{
    time_t t = time(NULL);
    tm lt = *localtime(&t);
    day = lt.tm_mday;
    mon = lt.tm_mon + 1;
    year = lt.tm_year + 1900;
    if(dateOnly) {
        hour = min = 0;
    } else {
        hour = lt.tm_hour;
        min = lt.tm_min;
    }
}
```

### Unique Date and Time integer value retrieval

The following function assigns a unique integer value to a date-time value. Using this value you can compare two dates. (i.e if the unique value of one date is greater than the unique value of another date, then the first date is larger than the second one)

## Project: Simple Point Of Sale System

### Latest Release

#### Milestones

#### Milestones due dates

#### Final project mark and due dates

#### Citation, Sources

See below for details:

For work that is done entirely by you (ONLY YOU)

For work that is done partially by you.

If you have helped someone with your code

#### Compiling and Testing Your Program

Adding alias to

`.bash_profile` on matrix

Project Implementation notes:  
*Very Important, read carefully*

#### The Point Of Sale system

#### Milestone 1

#### The PosApp Module

The features of the POS system

#### Implementation

The PosApp class

menu

```
int uniqueDateValue(int year, int mon, int day, int hour, int min) {
    return year * 535680 + mon * 44640 + day * 1440 + hour * 60 + min;
}
```

#### Number of days in the month based on leap year calculation

To find out what is the maximum valid value for a calendar day in a month you can use this function. This function gets the year and the month and gives you the correct value for the number of days in that month (considering the leap year).

```
int daysOfMonth(int year, int month){
    int days[] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31, -1 };
    int mon = month >= 1 && month <= 12 ? month : 13;
    mon--;
    return days[mon] + int((mon == 1) * ((year % 4 == 0) && (year % 100 != 0)) || (
```

#### Date Construction

- By default date is created and set to the current system date and will be set not to be date-only (it will include the hour and the minute)
- A date can get created by receiving the date; (year, month and day values only). In this case, the Date object will be set to date-only mode. After the attributes are set, they are **validated**.
- A date can get created using date and time; (hour, month, day, hour and minute). In this case, the Date object will not be set to date-only mode. After the attributes are set, they are **validated**.

#### Comparison Operator Overloads

Overload all the comparison operator overloads to compare two date objects and return a boolean.

(==, !=, <, >, <=, >=)

Make sure the references and methods are constant if they don't change the state of their objects and classes.

#### dateOnly Modifier method



## Project: Simple Point Of Sale System

### Latest Release

#### Milestones

#### Milestones due dates

#### Final project mark and due dates

#### Citation, Sources

See below for details:

For work that is done entirely by you (ONLY YOU)

For work that is done partially by you.

If you have helped someone with your code

#### Compiling and Testing Your Program

Adding alias to

`.bash_profile` on matrix

Project Implementation notes:  
*Very Important, read carefully*

## The Point Of Sale system

### Milestone 1

#### The PosApp Module

The features of the POS system

#### Implementation

The PosApp class

menu

Create a method called `dateOnly` that receives a boolean and returns the reference of the current date class.

This method should set the date-only flag to the value received from the argument of the function. However, if the function argument is "true" which sets the object to date only, then the hour and minute attributes of the date should be set to zero.

In any case, the reference of the current object is returned.

## boolean type conversion operator

Overload the bool type conversion operator and return the opposite of the state of the error attribute.

## the error query

Create a method called `error` that returns a constant reference of the error attribute object. This query does not change the state of the Date class.

## Other Methods and operation

Implement and add any other method or operations you find necessary to accomplish the above tasks (make them private if they don't need to be publicly accessible)

## ostream insertion operator overload

Overload the insertion operator to insert a Date into the ostream object. the result of the insertion should be as follows:

### If the Date is not in an erroneous state

If the Date is in the date-only mode, print the date in the following format: YYYY/MM/DD

If the Date is not in the date-only mode, print the date and time in the following format: YYYY/MM/DD, HH:MM

### if the Date is in an erroneous state

Insert the error attribute into ostream to print the error message held in the error attribute then print the date and time exactly as shown before but surrounding them with parenthesis.

```
Date A(2023,10,20);
Date B(2023,10,20,14,30);
Date C(2023,10,20,26,72);
```

## Project: Simple Point Of Sale System

### Latest Release

#### Milestones

#### Milestones due dates

#### Final project mark and due dates

#### Citation, Sources

See below for details:

For work that is done entirely by you (ONLY YOU)

For work that is done partially by you.

If you have helped someone with your code

#### Compiling and Testing Your Program

Adding alias to

`.bash_profile` on matrix

#### Project Implementation notes:

*Very Important, read carefully*

## The Point Of Sale system

### Milestone 1

#### The PosApp Module

The features of the POS system

#### Implementation

The PosApp class

menu

```
cout << A << endl << B << endl << C << endl;
/* output:
2023/10/20
2023/10/20, 14:30
Invalid Hour(2023/10/20, 26:72)
*/
```

DO NOT use friend to implement this. implement member functions to access the data instead.

### istream extraction operator overload

Overload the extraction operator to read a date from istream in the same format it is printed.

First, clear the error object to make sure the date is not in an erroneous state and set all the date and time attributes to zero.

Then read the date values by reading the five integers (if date and time) or the three integers (if date only), and ignoring the delimiters in the data entry.

Note that you do not need to verify the delimiters to be exactly what the printed format is.

For example, the following three date Entries are all valid:

```
2023/10/20, 26:72 <<- date and time
2023-10-20*26-72 <<- date and time
2023_10_20 <<- date only
```

After reading each integer, check the status of istream, if it is not in a fail state, ignore the next character and read the next integer. If the istream is in a fail state, set the error object to the corresponding error message (see below) and skip the rest of the entries. If all the integers were read successfully [validate](#) the date values.

#### Error messages for istream failure

```
"Cannot read year entry"
"Cannot read month entry"
"Cannot read day entry"
```

## Project: Simple Point Of Sale System

### Latest Release

### Milestones

### Milestones due dates

### Final project mark and due dates

### Citation, Sources

See below for details:

For work that is done entirely by you (ONLY YOU)

For work that is done partially by you.

If you have helped someone with your code

### Compiling and Testing Your Program

Adding alias to

`.bash_profile` on matrix

Project Implementation notes:  
*Very Important, read carefully*

## The Point Of Sale system

### Milestone 1

#### The PosApp Module

The features of the POS system

Implementation

The PosApp class

menu

"Cannot read hour entry"

"Cannot read minute entry"

Note if the Date is in read-only mode, after reading the three date integers, set hour and minute to zero.

In any case, at the end return the reference of istream.

## tester programs

Check each state of your program with the following testers. The main tester at submission combines all these tests.

- [01-ErrorTester.cpp - output](#)
- [02-constantValueTests.cpp - output](#)
- [03-DateConstructorTests.cpp - output](#)
- [04-DateLogicalOperators.cpp - output](#)
- [05-DateValidation.cpp - output](#)

## MS2 Submission

If you would like to successfully complete the project and be on time, **start early** and try to meet all the due dates of the milestones.

Upload your source code and the tester programs to your `matrix` account. Compile and run your code using the `g++` compiler [as shown in the introduction](#) 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 2??/prj/m2
```

and follow the instructions.

- 2?? is replaced with your subject code

### The submit program's options:

## Project: Simple Point Of Sale System

### Latest Release

#### Milestones

#### Milestones due dates

#### Final project mark and due dates

#### Citation, Sources

See below for details:

For work that is done entirely by you (ONLY YOU)

For work that is done partially by you.

If you have helped someone with your code

#### Compiling and Testing Your Program

Adding alias to

`.bash_profile` on matrix

#### Project Implementation notes:

*Very Important, read carefully*

### The Point Of Sale system

#### Milestone 1

##### The PosApp Module

The features of the POS system

##### Implementation

The PosApp class

menu

`~prof_name.prof_lastname/submit DeliverableName [-submission options]<ENTER>`  
`[-submission option]` acceptable values:

`"-due":`

Shows due dates only

This option cannot be used `in` combination with any other option.

`"-skip_spaces":`

Do the submission regardless of incorrect horizontal spacing.

This option may attract penalty.

`"-skip_blank_lines":`

Do the submission regardless of incorrect vertical spacing.

This option may attract penalty.

`"-feedback":`

Check the program execution without submission.

## Back to milestones

# Milestone 3



under construction

## MS3 Submission

If you would like to successfully complete the project and be on time, **start early** and try to meet all the due dates of the milestones.

Upload your source code and the tester program to your `matrix` account. Compile and run your code using the `g++` compiler [as shown in the introduction](#) 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 2??/prj/m3
```

and follow the instructions.

## Project: Simple Point Of Sale System

### Latest Release

#### Milestones

#### Milestones due dates

#### Final project mark and due dates

#### Citation, Sources

See below for details:

For work that is done entirely by you (ONLY YOU)

For work that is done partially by you.

If you have helped someone with your code

#### Compiling and Testing Your Program

Adding alias to

`.bash_profile` on matrix

Project Implementation notes:  
*Very Important, read carefully*

## The Point Of Sale system

### Milestone 1

#### The PosApp Module

The features of the POS system

#### Implementation

The PosApp class

menu

- 2?? is replaced with your subject code

## The submit program's options:

```
~prof_name.prof_lastname/submit DeliverableName [-submission options]<ENTER>
```

[-submission option] acceptable values:

**"-due":**

Shows due dates only

This option cannot be used **in** combination with any other option.

**"-skip\_spaces":**

Do the submission regardless of incorrect horizontal spacing.

This option may attract penalty.

**"-skip\_blank\_lines":**

Do the submission regardless of incorrect vertical spacing.

This option may attract penalty.

**"-feedback":**

Check the program execution without submission.

## Back to milestones

# Milestone 4

 under construction

## MS4 Submission

If you would like to successfully complete the project and be on time, **start early** and try to meet all the due dates of the milestones.

Upload your source code and the tester program to your `matrix` account. Compile and run your code using the `g++` compiler [as shown in the introduction](#) and make sure that everything works properly.

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

## Project: Simple Point Of Sale System

### Latest Release

#### Milestones

#### Milestones due dates

#### Final project mark and due dates

#### Citation, Sources

See below for details:

For work that is done entirely by you (ONLY YOU)

For work that is done partially by you.

If you have helped someone with your code

#### Compiling and Testing Your Program

Adding alias to

`.bash_profile` on matrix

Project Implementation notes:  
*Very Important, read carefully*

#### The Point Of Sale system

#### Milestone 1

#### The PosApp Module

The features of the POS system

#### Implementation

The PosApp class

menu

```
~profname.proflastname/submit 2??/prj/m4
```

and follow the instructions.

- 2?? is replaced with your subject code

### The submit program's options:

```
~prof_name.prof_lastname/submit DeliverableName [-submission options]<ENTER>
```

[-submission option] acceptable values:

**"-due":**

Shows due dates only

This option cannot be used in combination with any other option.

**"-skip\_spaces":**

Do the submission regardless of incorrect horizontal spacing.

This option may attract penalty.

**"-skip\_blank\_lines":**

Do the submission regardless of incorrect vertical spacing.

This option may attract penalty.

**"-feedback":**

Check the program execution without submission.

## Back to milestones

# Milestone 5



under construction

## MS51 submission test



under construction

## Data entry

## Project: Simple Point Of Sale System

### Latest Release

#### Milestones

#### Milestones due dates

#### Final project mark and due dates

#### Citation, Sources

See below for details:

For work that is done entirely by you (ONLY YOU)

For work that is done partially by you.

If you have helped someone with your code

#### Compiling and Testing Your Program

Adding alias to  
`.bash_profile` on matrix

Project Implementation notes:  
*Very Important, read carefully*

#### The Point Of Sale system

#### Milestone 1

##### The PosApp Module

The features of the POS system

##### Implementation

The PosApp class

menu

## Expected outcome

### reflection

Create a file called `reflect.txt` and add the following:

- Your Citation if you have any borrowed code in your project.
- Any additional (extra) work done that needs your professor's attention.
- Your overall reflection on the project and work done in the 5 milestones.

### MS51 Submission command

```
~profname.proflastname/submit 2??/prj/m51
```

### MS52 submission test

 under construction

### Data entry

### Expected outcome

### MS52 Submission command

```
~profname.proflastname/submit 2??/prj/m52
```

### Back to milestones

### MS53 submission test

 under construction

### Data entry



## Project: Simple Point Of Sale System

### Latest Release

#### Milestones

#### Milestones due dates

#### Final project mark and due dates

#### Citation, Sources

See below for details:

For work that is done entirely by you (ONLY YOU)

For work that is done partially by you.

If you have helped someone with your code

#### Compiling and Testing Your Program

Adding alias to  
.`bash_profile` on matrix

Project Implementation notes:  
*Very Important, read carefully*

#### The Point Of Sale system

#### Milestone 1

#### The PosApp Module

The features of the POS system

#### Implementation

The PosApp class

`menu`


## Expected outcome

### MS53 Submission command

```
~profname.proflastname/submit 2??/prj/m53
```

## Back to milestones

### MS54 submission test

 under construction

## Data entry

## Expected outcome

`m54-correct-output.txt`

### MS54 Submission command

```
~profname.proflastname/submit 2??/prj/m54
```

## Back to milestones

### MS55 submission test

## Data entry

## Expected outcome

### MS55 Submission command

```
~profname.proflastname/submit 2??/prj/m55
```

## Project: Simple Point Of Sale System

[Latest Release](#)

[Milestones](#)

[Milestones due dates](#)

[Final project mark and due dates](#)

[Citation, Sources](#)

See below for details:

For work that is done entirely  
by you (ONLY YOU)

For work that is done partially  
by you.

If you have helped someone  
with your code

[Compiling and Testing Your  
Program](#)

[Adding alias to  
.bash\\_profile on matrix](#)

[Project Implementation notes:  
\*Very Important, read carefully\*](#)

[The Point Of Sale system](#)

[Milestone 1](#)

[The PosApp Module](#)

[The features of the POS  
system](#)

[Implementation](#)

[The PosApp class](#)

[menu](#)

## Back to milestones