CCPROG3 Machine Project Specifications
1st Term, AY 2025-2026

Supermarket Simulator 🛒

Introduction

A supermarket is an establishment that pretty much everyone has been to—they've got everything a person needs from fruits to vegetables to canned goods to batteries. Because of its wide variety of products, a supermarket can be quite difficult to navigate for some people; they end up either getting lost or buying too much (or both). To make the experience easier, perhaps a simulation program of a supermarket can make it easier for them. I am running out of sentences to introduce this project. It's a supermarket come on guys you all know what it is (unless you live under a rock), no need for further introduction. Your task is to create a simulation of a supermarket facility, but with you as the sole shopper in the store (because it would be too complicated to add more shoppers for our course). Let's get right into the specifications.

Products

Of course, the heart of the supermarket is the products that it sells. All products have a unique serial number, a name, and a price. The serial number of a product is 8 characters long, where the first 3 characters depend on the product type, and the rest are digits that are unique per product.

Products that haven't been chosen by the shopper are always contained in a place to display them. Each product type may only be displayed in the specified display amenity. For example, it is not permissible to put fruits anywhere other than on a table.

The supermarket sells the following product types:

Product Type	Serial Number Format	Examples	Display
Fruit	FRUXXXXX	Apples, oranges, grapes	Table
Vegetable	VEGXXXXX	Cabbage, lettuce, broccoli	Table
Milk	MLKXXXXX	Fresh, soy, almond	Refrigerator
Frozen Food	FRZXXXXX	Hotdog, chicken nuggets, tocino	Refrigerator
Cheese	CHSXXXXX	Sliced, keso de bola, mozzarella	Refrigerator
Chicken	CHKXXXXX	Thigh fillet, breast fillet, ground	Chilled counter

Beef	BEFXXXXX	Rib, shank, ground	Chilled counter
Seafood	SEAXXXXX	Tilapia, sugpo, squid	Chilled counter
Bread	BRDXXXXX	Baguette, croissant, bagel	Table
Cereal	CERXXXXX	Oatmeal, barley, quinoa	Shelf
Noodles	NDLXXXXX	Instant noodles, ramen, miswa	Shelf
Snacks	SNKXXXXX	Candies, junk food, cookies	Shelf
Canned Goods	CANXXXXX	Canned tuna, sardines, soup	Shelf
Condiments	CONXXXXX	Salt, pepper, paprika	Shelf
Eggs	EGGXXXXX	Brown, quail, free-range	Table
Soft drink	SFTXXXXX	Sparkling water, cola, soda	Shelf
Juice	JUCXXXXX	Orange, pineapple, apple	Shelf
Alcohol	ALCXXXXX	Beer, vodka, soju	Shelf
Cleaning Agents	CLEXXXXX	Detergent, bleach, dish soap	Shelf
Home Essentials	HOMXXXXX	Broom, mop, plunger	Shelf
Hair Care	HARXXXXX	Shampoo, conditioner, hair wax	Shelf
Body Care	BODXXXXX	Soap, body wash, shower gel	Shelf
Dental Care	DENXXXXX	Toothpaste, toothbrush, dental floss	Shelf
Clothes	CLOXXXXX	Shirts, shorts, pants	Shelf
Stationery	STNXXXXX	Paper, tape, pencils	Shelf
Pet Food	PETXXXXX	Cat food, dog food, bird seed	Shelf

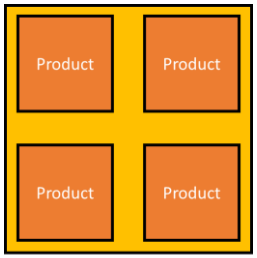
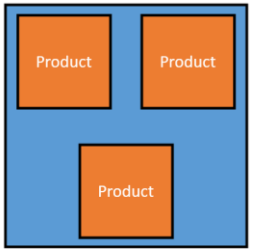
Restrictions and Discounts

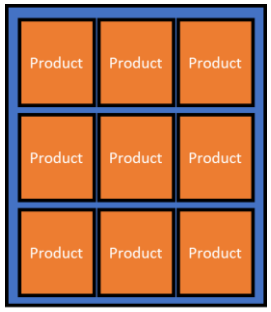
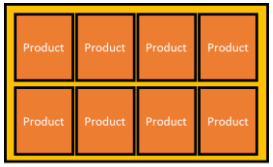
Depending on the age of the shopper, restrictions and discounts for certain product types for certain ages exist in the supermarket.

Shoppers under the age of 18 are not allowed at all to get Alcohol and Cleaning Agent products from the displays. Meanwhile, shoppers 60 and above are entitled to a Senior Citizen's Discount on all consumable products (e.g., items that can be eaten or drank) *except for* Alcohol product types. The Senior Citizen's discount is 20% off for food and 10% off for beverages.

Displays

Products may be displayed in one of the 4 types of displays listed below. Note that the real-world examples are quite complicated and large; the requirements of displays in the simulation have been simplified and scaled down.

Name	Description	Representation	Real-world Example
Table	Displays products arranged in a single horizontal layer on a flat surface. A single table can contain 4 products.	 Table	 Source: https://www.tripadvisor.com/LocationPhotoDirectLink-g312692-d4367351-i138428039-SM_City_Marikina-Marikina_Metro_Manila_Luzon.html
Chilled counter	Displays products arranged in a single horizontal layer on a flat surface that is constantly chilled. Each chilled counter can contain 3 products.	 Chilled counter	 Source: https://www.youtube.com/watch?v=Mx4tNnAlmh4

Refrigerator	Displays products arranged in 3 tiers inside a climate-controlled appliance. Each tier can contain 3 products.	 Refrigerator	 Source: https://www.youtube.com/watch?v=xkqDyk0-lNQ
Shelf	Displays products arranged in 2 tiers in a shelf cabinet. Each tier can contain 4 products.	 Shelf	 Source: https://www.youtube.com/watch?v=-cYsubh9mu4

Address

Displays also have an address to identify their location in the supermarket. The address is composed of the following:

- What floor the display is in (i.e., GF or 2F),
- What grouping (i.e., isle or wall) the display is in, and
- A unique number.

An example of an address is “2F, Aisle 21, Shelf 3”. No two displays have the same two addresses. Uniqueness is determined by *all* the components of the address. For example, “2F, Aisle 21, Shelf 3” is distinct from “GF, Aisle 2, Shelf 3” and “2F, Aisle 21, Shelf 4” and “2F, Aisle 14, Shelf 3”.

Shopper

The shopper will be your avatar when you go shopping in the supermarket simulation. The shopper has a name to identify it as well as an age.

Equipment

To allow shoppers to collect many items for checkout, shoppers should get equipment to store the products in. Available equipment are baskets and carts, which may both be retrieved from their respective stations.

Name	Product Capacity
Basket	Can contain a maximum of 15 products.
Cart	Can contain a maximum of 30 products.

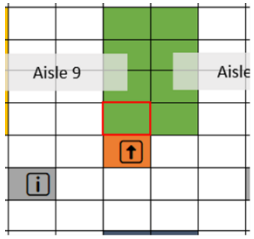
Shoppers may have with them at most one equipment. Without equipment, shoppers may only carry with them a maximum of 2 products in their person (i.e., hand-carried).

Representation and Movement

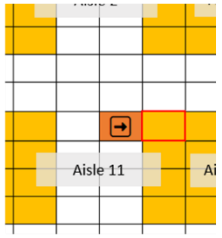
Shoppers, regardless of whether or not they have equipment with them, fill a single tile on the supermarket map. Shoppers can move one tile at a time in one of the four cardinal directions (i.e., north ⬆, east ➡, west ⬅, or south ⬇) as long as the tile is passable.

Vision

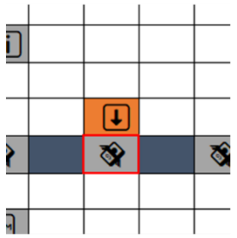
Aside from movement, the shopper can also face in one of the cardinal directions. Where the shopper faces determines the shopper's *vision*. If a shopper is facing an amenity, it can interact with it. To the right are examples of how the shopper's vision works, with the direction of vision is represented using arrows. Where the shopper faces is independent from the shopper's movement (i.e., they don't influence each other, so a shopper can move to the east while facing north).

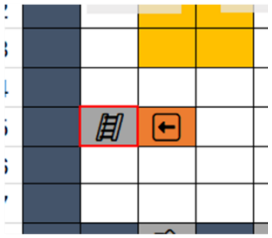


The shopper can see the table display and its products to its north. Hence, the shopper can interact with the table.



The shopper can see the shelf display and its products to its east. Hence, the shopper can interact with the shelf.


The shopper can see the checkout service to its south. Hence, the shopper can interact with the checkout counter.


The shopper can see the stairs to its west. Hence, the shopper can interact with the stairs (and go up or down to the next level).

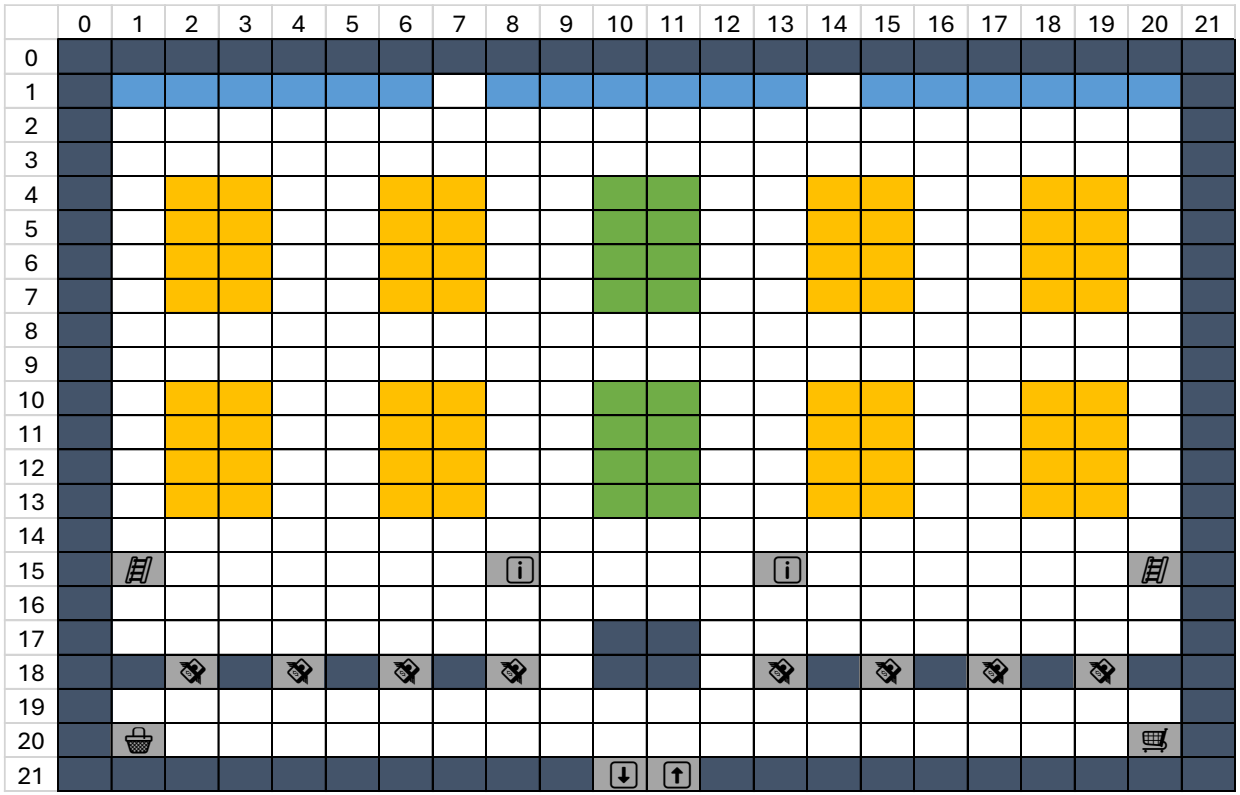
Supermarket

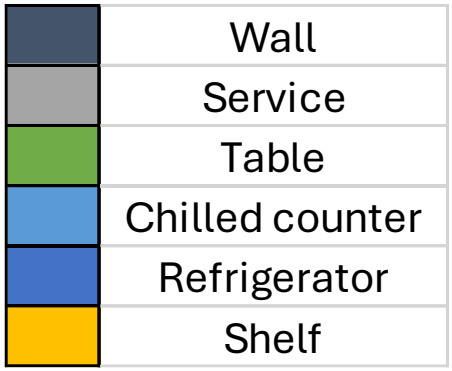
The supermarket you are going to simulate has two floors, as explained below.

Floors

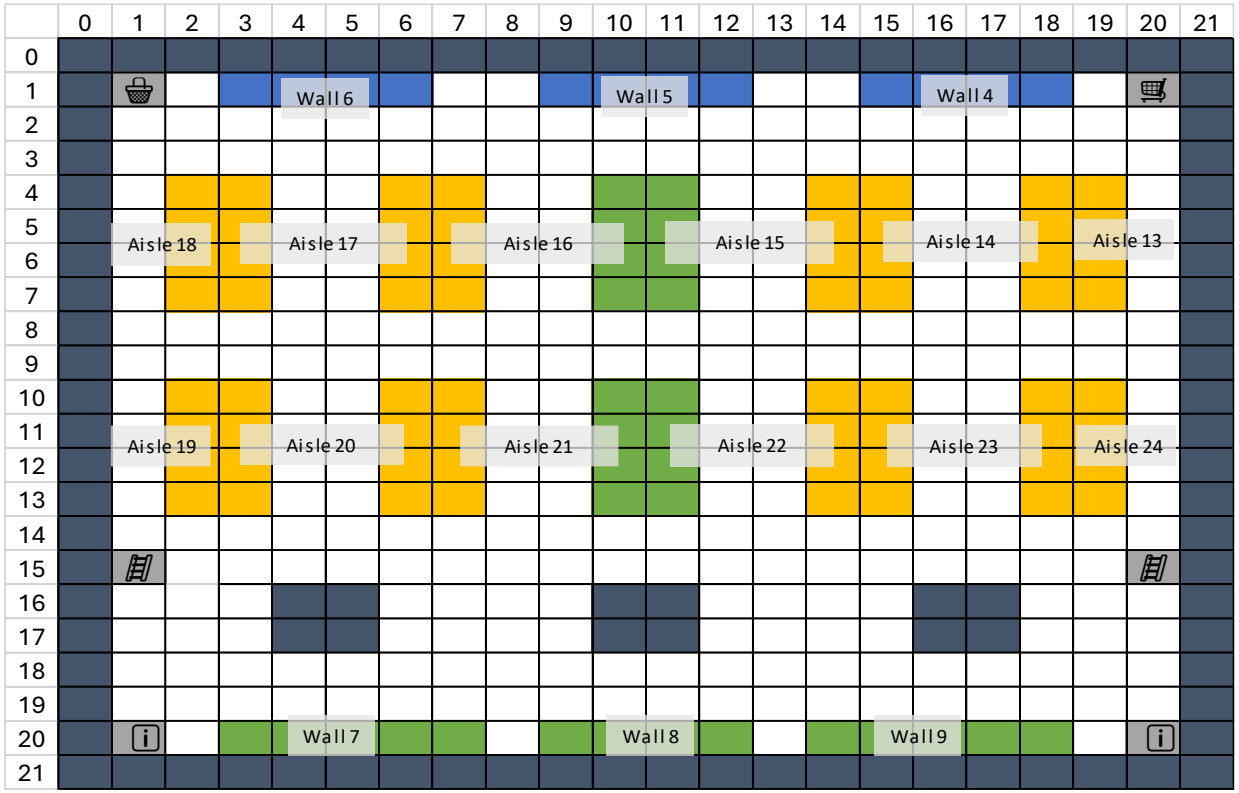
Each floor is represented by a map which is composed of a grid of 22x22 tiles. Each tile may contain one of the amenities represented by colors given in the legend to the right. It is also possible for a tile to contain no amenities at all. This denotes a free space—shoppers may freely move around these free spaces.

Ground Floor (GF)





Second Floor (2F)



Note: The stairs at Row 1, Column 15 and Row 20, Column 15 are connected with their respective counterpart portals at the other floor. That is, if a shopper enters the stairs at Row 1, Column 15 at the ground floor, the shopper should emerge at Row 1, Column 15 at the second floor.

Amenities

Amenities are any kind of facility that the shopper may interact with in the supermarket map. These include the displays mentioned earlier. Shoppers may not pass through amenities on the map, *except for services*.

Wall

Walls are used to create impassable boundaries in areas of the map. They offer nothing other than being a physical barrier.

Service

Services represent facilities in the supermarket that allow the shopper to perform tasks. A shopper can choose to interact with services whenever they are within a shopper's vision. These actions of services when interacted with are represented in the table below.

Service	Description	Action When Interacted with By the Shopper
Entrance	Shoppers in the simulation spawn here.	None. Once a shopper is spawned, the entrance does not have any tasks to offer to a shopper.
Cart station	Allows shoppers to get or return carts.	The cart station should allow shoppers who: • Do not have any equipment, and • Do not have any hand-carried products, and • Have not yet checked out to get a cart. For simplicity, it's assumed that the cart station has an unlimited supply of carts. The cart station should also allow shoppers to return their carts for any reason as long as the carts are empty.
Basket station	Allows shoppers to get or return baskets.	The basket station should allow shoppers who: • Do not have any equipment, and • Do not have any hand-carried products, and • Have not yet checked out to get a basket. For simplicity, it's assumed that the basket station has an unlimited supply of baskets. The basket station should also allow shoppers to return their baskets for any reason as long as the baskets are empty.
Product search	Allows shoppers to search for products in the supermarket.	The shopper is asked for the name of the product to search. The product search facility then shows the addresses of all displays where the product is in (or none if the product does not exist anywhere in the supermarket).
Stairs	Allows shoppers to ascend or descend to the other floor.	The shopper is transported to the other end of the stair portal (which is on another floor). More precisely, a shopper despawns from the floor the shopper is in, the interface is changed such that it now displays the other floor, and then the shopper is spawned at the same grid coordinates of the stair portal it just left, but by this time the shopper is now on the other floor.

Checkout counter	Where shoppers pay for their products.	This counter allows shoppers to check their products out. When checking out, a receipt should be saved **as a text file** containing a list of all products bought, their serial numbers, each product's price, the total price, and the discounted total price if eligible. If the shopper has any equipment, the checkout counter takes it from the shopper. At this point, the shopper is now free to leave the supermarket. Checkout counters should not service shoppers who try to check out with no products with them.
Exit	Allows shoppers to leave the simulation.	An exit should allow shoppers who: • Have checked their products out (if any), and • Do not have equipment with them. To despawn from the supermarket, marking the end of the simulation. A shopper who did not check anything out may also freely leave the supermarket as long has the shopper had no equipment with them (i.e., shoppers can't take equipment out of the supermarket).

Displays

As explained previously, displays are the amenities where products are displayed on (or in). A shopper can choose to interact with displays whenever they are within a shopper's vision.

A shopper can remove a product from any display (for checkout later) as long as the shopper is not restricted to do so. For multi-tiered displays (i.e., refrigerators and shelves), shoppers can choose from which tier in the display to remove products from.

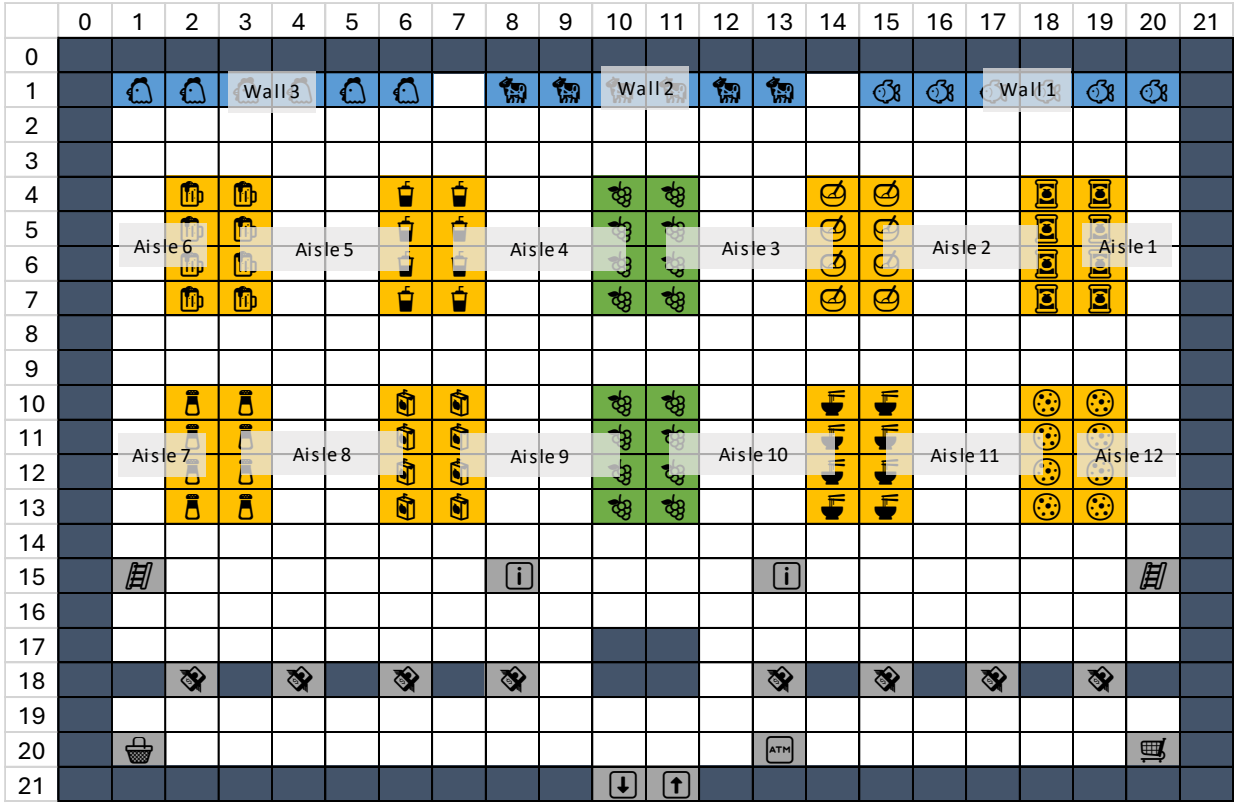
The opposite is also allowable: a shopper can return a product back to a display as long as the product is allowed to be on that display and that there still is space on that display. For multi-tiered displays, a product may be returned to a tier within the display as long as there is still space within that tier.

Simulation

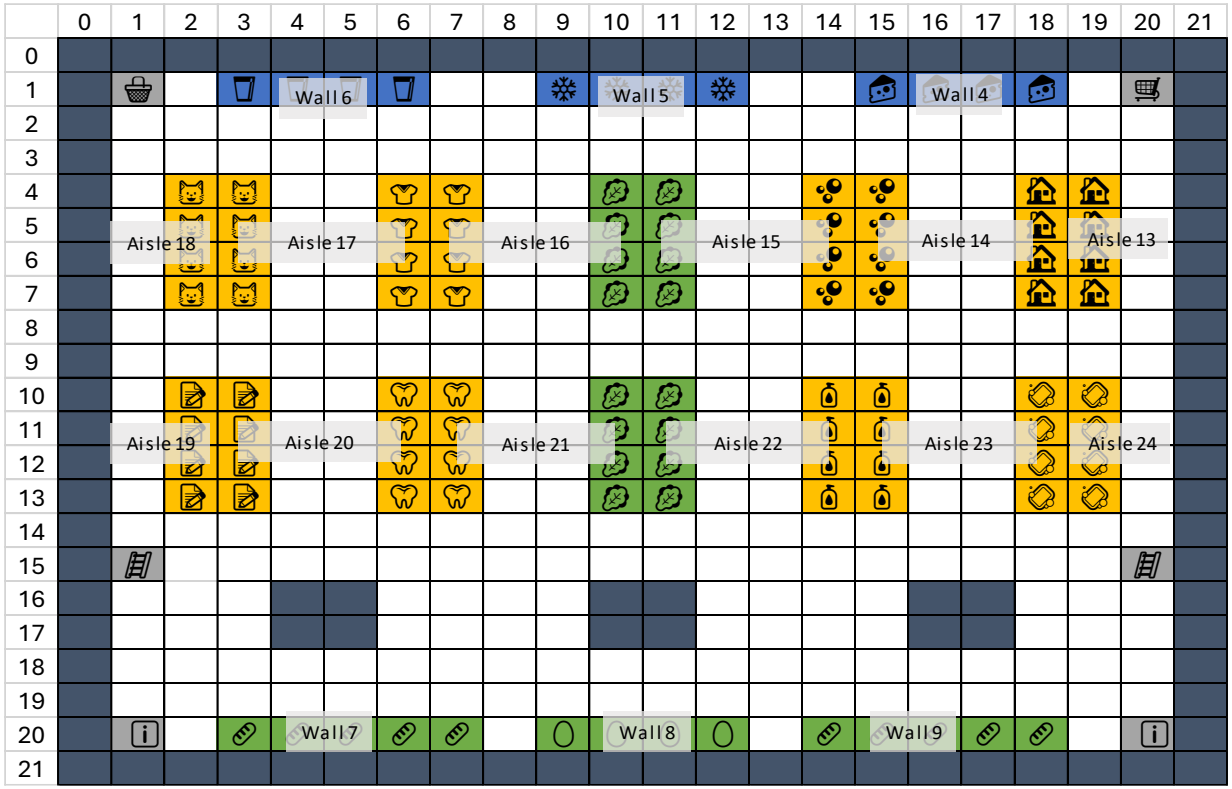
Initialization

In the beginning of the simulation, the shopper's name and age is asked. After this, the shopper spawns at the entrance tile of the supermarket. The supermarket's floors should have the following amenities at their exact positions specified below, with each display stocked with products of the following product types for each floor:

Ground Floor Products



Second Floor Products



Initial Products

For each product type, please create at least 3 products with the name and price up to your creativity. For instance, you should create at least 3 examples of cereal products (e.g., Java Cereal, OOPsie Oaties, Barley Bytes). It's okay to repeat products in the displays (this always happens in the real world) as long as I see at least 3 unique products for each of the product types specified in the Products section earlier in this document.

Initial Display Contents

Each display has to be stocked with products of the required product type on initialization. The products can be randomly generated and assigned to random places in the proper displays. The display need not be full of products – you may leave displays with some empty gaps.

Navigation

From the entrance, you may control the shopper's movement using the WASD keys* (W – north, A – west, S – south, D – east), and the shopper's vision using the IJKL keys* (I – north, J – west, K – south, L – east). Make sure not to pass through amenities that are not passable, as well as to not allow moving out the bounds of the map.

From this point on, the shopper can now navigate the supermarket and interact with the amenities within a shopper's vision. The actions when a shopper interacts with an amenity are detailed in the previous sections. To interact with an amenity, use the Space button* (in MCO1) or click on the amenity* (MCO2).

View Chosen Products

At any point in the simulation, you may click the V key* (in MCO1) or press a button in the GUI (in MCO2) to see the contents of your cart or basket (or your hands, if you have no equipment). In this view, the equipment chosen by the shopper is shown, plus the following details:

- All unique products chosen (i.e., if I bought 7 Barley Bytes, then there should only be just 1 entry for Barley Bytes),
- The quantity chosen of each unique product (i.e., in the example above, the quantity should be 7),
- The total price of each unique product (i.e., in the example above, if each Barley Bytes is 100 PHP, the value here should be 700 PHP).

** You are allowed to use other keyboard or user interface controls if you can think of one which works better with your product.*

Ending

When a shopper leaves the supermarket for any reason, the user is asked whether to restart the simulation or not. If the user chooses to restart, then the simulation starts from the beginning, with all the proper initial values reset. If the user chooses to end the simulation, then the simulation should close.

Milestones (see Deliverables section)

a. **MCO1 – due 9:00 pm, October 24, 2025 (F)**
   1. UML Class Diagrams for **model classes only** (i.e., excluding classes used for UI, the driver class, etc.), but not necessarily including the excluded features below. Use of inheritance is optional for MCO1 if you're willing to do some advanced studies.
   2. Classes containing the implementation of all features and functionalities in the specification above **except for the following, which will only be required in MCO2**:
      a. The second floor of the supermarket
      b. The stairs amenity
      c. The following product types:
         i. Pet Food
         ii. Clothes
         iii. Stationery
         iv. Dental Care
         v. Cleaning Agents
         vi. Household Care
         vii. Hair Care
         viii. Body Care
         ix. Vegetables
         x. Milk
         xi. Frozen Foods
         xii. Cheese
         xiii. Bread
         xiv. Eggs
   3. Driver class to test the basic application. No GUI display is expected. However, displays must be made in the console.

b. **MCO2 – due 12:00 nn, November 24, 2026 (M)**
   1. UML Class Diagrams for **model classes only**, with mandatory use of inheritance.
   2. Comprehensive, canvas-drawn GUI (i.e., don't just plop your console application in a GUI window and call it a day) with mouse-controlled inputs. **Recommended GUI library**: explore **JavaFX** and its associated tools (e.g., Scene Builder)
   3. Complete implementation of the entire simulation.
   4. Using the Model-View-Controller (MVC) design pattern.

Deliverables

The deliverables for both MCOs include:

1. The design and implementation of the solution should:
   - Conform to the specifications described above
   - Exhibit proper object-based / object-oriented concepts, like encapsulation and information-hiding, etc.
   - **NOT** be derived or influenced from any use of Generative AI tools or applications
2. Signed declaration of original work (declaration of sources and citations may also be placed here)

    - See **Appendix A** for an example
3. Softcopy of the class diagram following UML notations (in pdf or png)
    - Kindly ensure that the diagram is easy to read and well structured
4. Javadoc-generated documentation for proponent-defined classes with pertinent information
5. Zip file containing the source code with proper internal documentation
    - The program must be written in Java
    - Include external libraries that were used (e.g., JavaFX).
6. Test script following the format indicated in **Appendix B**
    - In general, there should be at least 3 categories (as indicated in the description) of test cases per method (except for setters and getters).
    - There is no need to test user-defined methods which are ONLY for screen design (i.e., no computations/processing; just print/println).
7. **For MCO1 only**: A video demonstration of your program
    - While groups have the freedom to conduct their demonstration, a demo script will be provided closer to the due date to help with showing the expected functionalities.
    - The demonstration should also quickly explain key aspects of the program's design found in the group's class diagram
    - Please keep the demo as concise as possible and refrain from adding unnecessary information
8. Groups should back-up their projects. A softcopy of the final unmodified files (for each phase) should be sent to their own emails, apart from regular submissions of progress in AnimoSpace and/or Git.

Submission

All deliverables for the MCO are to be submitted via AnimoSpace. Submissions made in other venues will not be accepted. Please also make sure to take note of the deadlines specified on AnimoSpace. Late submissions will not be accepted.

Grading

For grading of the MCO, please refer to the MCO rubrics indicated in the syllabus.

Collaboration and Academic Honesty

This project is meant to be developed as a pair pfor both phases (i.e., MCO1 and MCO2) barring any reports of freeloading or decision by the pair to split. In exceptional cases, a student may be allowed by their instructor to work on the project alone; however, permission should be sought as collaboration is a key component of the learning experience. Under no circumstance will a group be allowed to work on the MCO with more than 2 members.

A student cannot discuss or ask about design or implementation with other persons, with the exception of the teacher and their groupmate. Questions about the project specifications should be raised in the Discussion page in AnimoSpace. Copying other people's work and/or working in

collaboration with other teams are not allowed and are punishable by a grade of 0.0 for the entire subject and a case may be filed with the Discipline Office. In short, do not risk it; the consequences are not worth the reward. Comply with the policies on collaboration and AI usage as discussed in the syllabus.

Documentation and Coding Standards

Do not forget to include internal documentation (comments) in your code. At the very least, there should be an introductory comment and a comment before every class and every method. This will be used later to generate the required External Documentation for your project via Javadoc. You may use an IDE or the appropriate command-based instructions to create the documentation, but it must be PROPERLY constructed.

Please note that you're not expected to provide comments for each and every line of code. A well-documented program also implies that coding standards are adhered to in such a way that they aid in the documentation of the code. Comments should be considered for more complex logic.

Bonus Points

Bonus points will only be awarded for MCO2. The above description of the program is the basic requirement. Any additional feature will be left to the creativity of the group. Bonus points would be awarded depending on the additional implemented features. These additional features could include:

- Allow shoppers to have **money**. When checking out, the shoppers should have enough money for all their items. Otherwise, the checkout counter should refuse to let the shoppers check out until they get enough money. Shoppers should get money from an ATM service somewhere in the supermarket map, which you should also add if you're implementing this bonus feature. I'll award up to 6 bonus points for this.
- Add a **spectator mode** to the simulation where multiple, automatically-generated shoppers find their way around the supermarket and check out on their own. This is very difficult, so I can award up to 10 bonus points for this if you manage to do it.
- Add a **storekeeper mode** to the simulation where you can manually choose what products to restock. I can award up to 8 bonus points for this.
- To encourage the **usage of version control** (such as Git), up to 4 bonus points may be awarded for this. While the usage of version control will not be taught in this course, you are encouraged to organize and store your code via a Git repository, like the services offered by GitHub. Utilizing some form of version control will make it easier for the members of the group to collaborate with each other and the commit history also helps in providing some form of accountability. Awarded points may vary based on how the group was able to leverage the usage of version control (e.g. small and often commits, descriptive commit comments, contributions from all members, branching).

Depending on the scale of the new feature, additional points will be awarded to the group. However, make sure that all the minimum requirements are completely and correctly met first; if this is not

the case then no additional points will be credited despite the additional features. To encourage the usage of version control, please note that a small portion of the bonus points for MCO2 will be the usage of version control. Please consider using version control as early as MCO1 to help with group collaboration.

Resources and Citations

All sources should have proper citations. Citations should be written using the APA format. Examples of APA-formatted citations can be seen in the References section of the syllabus. You're encouraged to use the declaration of original work document as the document to place the citations.

Further, this is to emphasize that you DO NOT need to create your own sprites (e.g., pictures of products) for the simulation. You can just use what's available on the Internet and just include these into your project, just make sure to cite your sources.

Demo

Demo for MCO1 is via a video submission. All members are expected to be present in the video demonstration and should have relatively equal parts in terms of the discussion. Any student who is not present during the demo will receive a zero for the phase.

In MCO2, demo is live and will include an individual demo problem. Schedule for the demo will generally be during class time, but there may be other schedules opened by the faculty in case there is not enough time to accommodate everyone. Sequence (of who goes first in the class to do the demo) is determined by the faculty. Thus, do not be absent or late during announced demo days. A student or a group who is not present during the demo or who cannot answer questions regarding the design and implementation of the submitted project convincingly will incur a grade of 0 for that project phase.

During the demo, it is expected that the program can be compiled successfully in the command prompt and will run. If the program does not run, the grade for that phase is 0. However, a running program with complete features may not necessarily get full credit, as implementation (i.e., code) will still be checked.

Other Notes

You are also required to create and use methods and classes whenever possible. Make sure to use Object-Based (for MCO1) and Object-Oriented (for MCO2) Programming concepts properly. No brute force solution.

Statements and methods not taught in class can be used in the implementation. However, these are left for the student to learn on his or her own.

Appendix A. Template for Declaration of Original Work

Declaration of Original Work

We/I, [Your Name(s)] of section [section], declare that the code, resources, and documents that we submitted for the [1st/2nd] phase of the major course output (MCO) for CCPROG3 are our own work and effort. We take full responsibility for the submission and understand the repercussions of committing academic dishonesty, as stated in the DLSU Student Handbook. We affirm that we have not used any unauthorized assistance or unfair means in completing this project.

[*In case your project uses resources, like images, that were not created by your group.*] We acknowledge the following external sources or references used in the development of this project:

1. Author. Year. Title. Publisher. Link.
2. Author. Year. Title. Publisher. Link.
3. Author. Year. Title. Publisher. Link.

By signing this declaration, we affirm the authenticity and originality of our work.

Signature and date	*Signature and date*
Student 1 Name ID number	Student 2 Name ID number

[*Note to students:*

1. *Do not submit documents where your signatures are easily accessible. Ideally, submit a flattened PDF to add a layer of security for your digital signatures*
2. *You may use the eSignature feature of Google Docs. See this template.*]

Appendix B. Example of Test Script Format

Class: MyClass						
Method	**#**	**Test Description**	**Sample Input Data**	**Expected Output**	**Actual Output**	**P/ F**
isPositive	1	Determines that a positive whole number is positive	74	true	true	P
	2	Determines that a positive floating point number is positive	6.112	true	true	P
	3	Determines that a negative whole number is not positive	-871	false	false	P
	4	Determines that a negative floating point number is not positive	-0.0067	false	false	P
	5	Determines that 0 is not positive	0	false	false	P