Workshop - 6

Workshop Value: 10 marks (4.375% of your final grade)

Please review the following documents:

1. Workshop **Grading Policies**
2. Workshop **Submission Procedures**
3. Workshop **Group Breakdown**

Workshop Overview

Vending machines now almost run without any human maintenance. They employ the "internet of things" (**IoT**) enabling real-time updates on stock levels, payments, and alerts for machine maintenance. Electronic payments use both swipe and "tap" technology for debit and credit cards and even accept cell phone payments using "near field communications" (**NFC**). No physical money is stored! Maintenance costs are drastically reduced with these enhancements over the older models as routine inventory and money pickups are eliminated. The only time a service provider needs to physically visit the machine, is for restocking inventory and addressing any general mechanical maintenance (which would be infrequent). **But how should this all work?**

Workshop Details

Applications of vending machines essentially have **two major logical components to define**:

1. **Hardware States** (physical interface)
2. **Software Logic** (main functional logic/controller)

States

Hardware states are triggered by the software layer that implement the overall system process. The system has at least 6 main states (or modes):

1. Power-On
2. Power-Off
3. **Idle/Ready** (stand-by: waiting for customer)
4. **Active** (customer interaction building an order)
5. **Payment** (final step in customer transaction)
6. **Cancel** (cancels the session at any time during states 4&5)

$\left.\right\}$ **You define these**

For each of these states, both the **hardware** and **software** components will have their own set of defined processes.

Data Structures

Use the following data structures in your solution:

Product

slotID	// Unique location slot ID (physical placement in the machine ex: "D8")
sku	// Unique product identifier
quantity	// Actual quantity available (physically in machine at the given slotID)
maxQuantity	// Maximum machine qty that can be stocked for the slotID
minQuantity	// Re-order when this qty is reached (based on: maxQuantity - quantity)
price	// Vending machine price to charge customer per unit
description	// Product name

*Note: The term "**Inventory**" is simply a **collection/array** of Product data*

<u>**Transaction**</u>

slotID	// Unique location slot ID (physical placement in the machine ex: "D8")
quantity	// Requested quantity
price	// Price per single unit quantity
description	// Product name

<u>Note</u>: The term *"**<u>Session</u>**" is simply a **collection/array** of <u>Transaction</u> data*

[Logic 1] <u>**Hardware States**</u> & <u>**Software Idle State**</u>

- Hardware components have only two possible states: "**enabled**" or "**disabled**"
- As the machine changes from one state to another, the various hardware components need to be initialized to the new state and should be set to complement the initial state of the software logic (example: disable controls that don't apply to the current state and only enable those that should be)
- The software logic will modify/change the hardware components as needed <u>after</u> the initialization of the hardware is set to the new state
- For <u>each</u> vending machine state, create the necessary initial hardware settings
  - Define each hardware component's state (***enabled*** or ***disabled***)
  - Each hardware component should be described in its own step/process
  - You should have 4-sub-processes in total (**one for each vending machine state**)
    <u>Hint</u>: The software logic layer will "call" the hardware processes when initializing to a new state
- The software idle state essentially waits until it is activated by a customer (be creative on what the machine can be doing during the idle state!)

[Logic 2] <u>**Product Selection**</u> (involves the **Active** and **Cancel** states)

- The active state creates a session and builds a series of transactions
- Slots are limited to a single letter (row) button followed by a single number (column) button sequence (ex: "A" button + "9" button)
- After selecting a slot (product), you can enter a quantity from 1-9 followed by the **Enter** key. The **Correct** button can be used to erase the quantity before the **Enter** key is pressed.
- A product selection can only become a transaction if the requested quantity for the item is in-stock (has inventory)
- Multiple transactions can be added to the session by entering a series of products one selection after another
- Itemized on-screen session transaction details should be displayed
- Using the **Pay** button will indicate the end of the transactional session
- Create <u>**two separate sub-processes**</u>: **Active** and **Cancel** (states) that cover the software logic involving the **creation of a session** with **transactions** (**product selection**), **entering item quantities**, **correcting inputs**, and **cancelling the session**.

[Logic 3] <u>**Acceptable Payments & Product Inventory**</u> (**Payment** and **Cancel** states)

- **1.** Credit card, **2.** Bank card **3.** NFC phone **4.** Vending machine phone application
- **$$ NO CASH $$**
- Following a successful payment, inventory must be adjusted in real-time (hint: local and remote data)
- Inventory minimum stock levels must be enforced (if the quantity in-stock reaches the minimum stock level set for that product's row & column location, more products should be ordered
- Dispense the items only after payment is successful
- Create two separately sub-processes: **Payment** and **Cancel** (states) that cover the software logic involving **payment**, **inventory adjustments**, **dispensing the items**, and **cancelling the session**.

[Group Solution] Main Process

- Create a "main" process that will apply the overall vending machine logic
- Include the hardware initialization sub processes
- Include all sub-processes that are needed to process product selections, payments, and inventory adjustments (with cancel logic)
- Hint: the vending machine should work continuously until it is manually shutdown (interrupt/override)

Your Task

Individual Logic Assignment

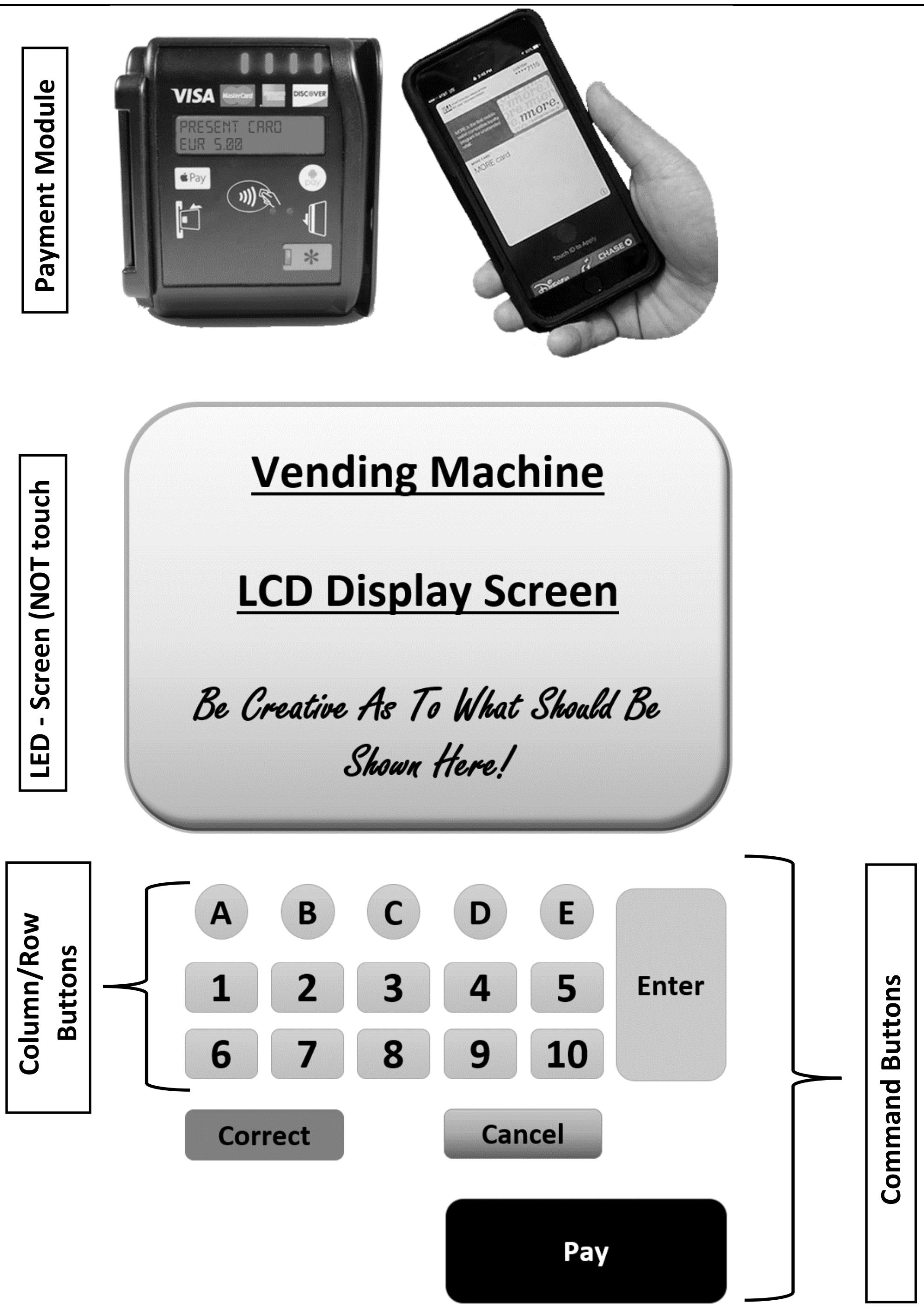
1. Determine your individual assigned logic part based on your member# (see **Group Breakdown** link at the beginning of this document)
2. Where applicable, apply the core components of the **computational thinking** approach to problem solving to help you synthesize a solution
3. Submit your individual assigned part to your professor (see **Submission Procedures** link at the beginning of this document)

Group Solution

1. In the week the workshop is scheduled, you will be working in your assigned sub-group. See **Group Breakdown** link at the beginning of this document for details on how the sub-groups are determined.
2. Please review what is expected as described in the **Grading Policies** link at the beginning of this document.
3. Submit your group solution to your professor (if you are handing in physical paper answers, follow the directions as set by your professor, otherwise, refer to the **Submission Procedures** link at the beginning of this document)

Presentation

Decide among yourselves which member among you in the **sub- group** will be doing a presentation. Priority should be given to those who have not yet done one. Refer to the **Grading Policies**, and **Submission Procedures** links for details on deadlines, expectations and how to submit your work.



Hardware Controls/Interface

• Payment Module:	Physical hardware for reading credit cards, debit, NFC, Tap etc.
• LCD Colour Screen:	10” wide X 6” high (10 cm X 15 cm)
• Column/Row buttons:	Column-letter buttons (A-E), row-number buttons (1 – 10)
• “Enter” button:	Adds a product selection to the session (transaction), or applies entered quantity
• “Correct” button:	Used to backspace an entry/quantity, NOT remove an already added item
• “Cancel” button:	Cancels the entire session and resets
• “Pay” button:	Triggers the payment process and finalizes the session