Workshop - 10

Workshop Value: 10 marks (4.4% of your final grade)

Learning Outcomes

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

- to decipher and identify a problem
- to analyze and decompose a problem
- to identify the required detailed steps to solve a **larger problem** using **modularity**
- to communicate the solution to fellow peers and non-technical business persons

Workshop Grading and Promotion Policy

Workshops for this course will be assessed using the following criteria:

- Workshops must be completed before the class time to be graded
- You must successfully complete 9 workshops (if more than 9 are completed, the best 9 will be used)
- Each student is expected to be a presenter of the workshop solution at least once by the end of the term
- Workshop solutions and presentations will be evaluated using the published workshop rubrics

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 main 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 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**

You define these states

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

Constraints

- **[Logic 1] Hardware States**
    - Hardware components have only two possible states: "**enabled**" or "**disabled**"
    - For each machine state, the various hardware components should be set to complement the software logic (example: disable controls that don't apply to the current state and only enable those that should be)
    - Logic 1 will show how the various hardware components are enabled or disabled at various times and what events cause them to be enabled or disabled.
- **[Logic 2] Product Selection**
    - Limited to a single letter (column) button followed by a single number (row) button sequence (ex: "A" button + "9" button)
    - After selecting a 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.
    - Products can only be added to a transaction if the requested quantity for the item is in-stock (has inventory)
    - Multiple products can be added to a single transaction by entering one selection after another.
    - Using the pay button will indicate the end of the session
- **[Logic 3] Product Inventory**
    - Following 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
- **[Logic 3] Accept Payment**
    - Credit card (tap/swipe)
    - Bank card (tap/swipe)
    - NFC phone payment
    - Phone application payment
    - **$$ NO CASH $$**
    - This logic will handle displaying the price, reading the card, transacting with the card company and accepting payments. It will also adjust inventory and order more if needed.

Hardware Controls/Interface (*see last page for illustrations*)

- 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

Data Structures

Apply the following data structures in your solution:

"Product" (*"Inventory" is simply a **collection/array** of "Product" data*)

slotID	// Unique location slot ID (where it is physically placed 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

"Transaction" (*"Session" is simply a **collection/array** of "Transaction" data*)

slotID	// Unique location slot ID (where it is physically placed in the machine ex: "D8")
quantity	// Requested quantity
price	// Price per single unit quantity
description	// Product name

Your Task

IMPORTANT: Divide the work up among your group members where possible to ensure you complete the requirements on-time. You will need to discuss the overall flow and overlapping logic periodically to ensure your solutions piece together seamlessly.

1. **Hardware**:

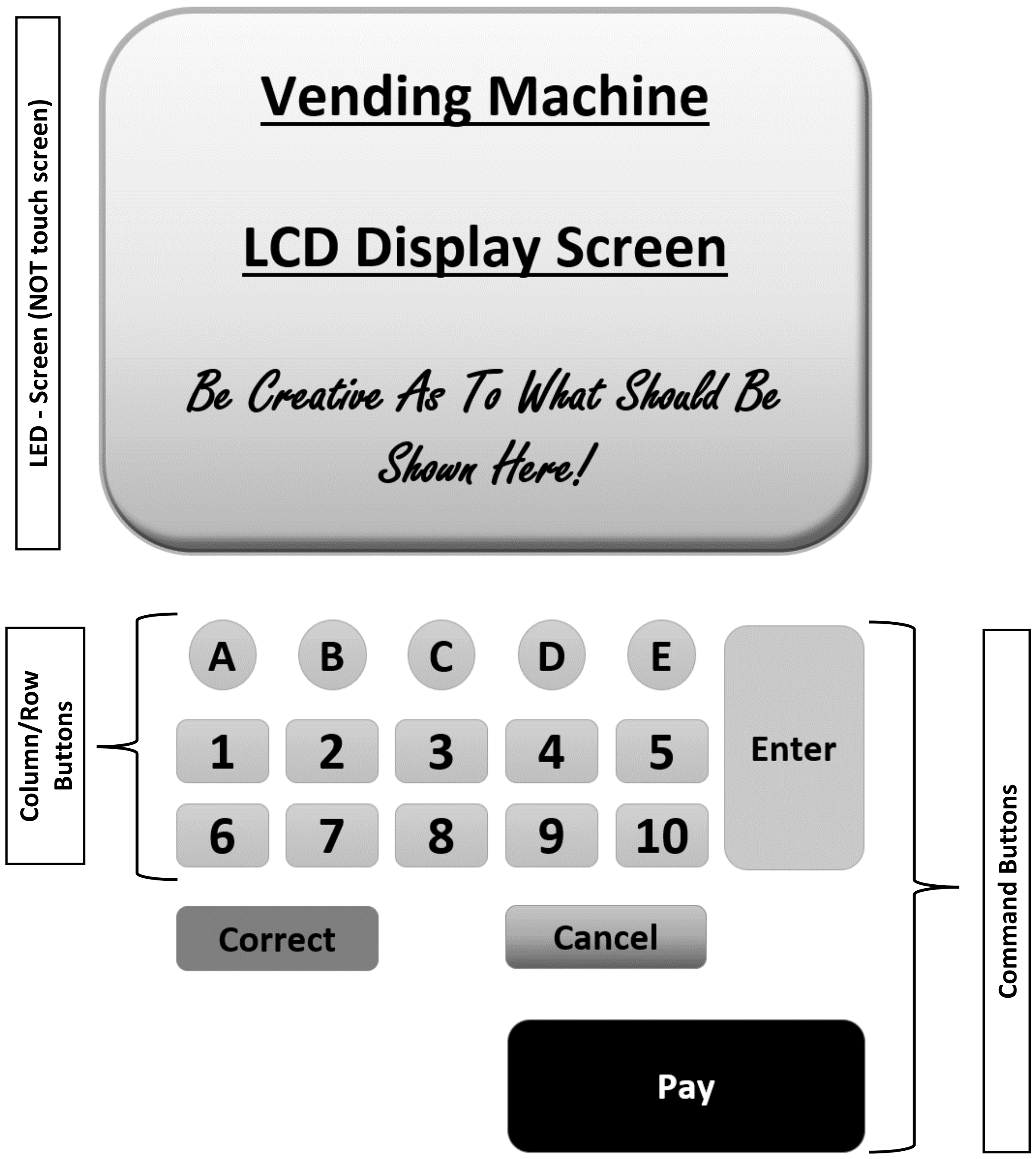
   - For each state, create the necessary ***FLOWCHARTS*** (pseudo code not required, even people in the pseudocode group will do a flowchart of this one)
     - Essentially, define what controls need to be ***enabled*** or ***disabled*** per state
     - You should have 4-flowcharts in total *(this should take no more than 15 minutes)*

   Hint: the software logic layer will "call" these processes as required when changing states

2. **Software**: For Logic 2 and Logic 3, define how the machine should work (***pseudo code*** or ***flowchart as assigned***)
   - You should have 2 separately defined pseudo code processes
   - You should have 2 separately illustrated flowcharts
   - Combined pseudo-code and flowchart that put Logic 1 and Logic 2 together.

3. **Present Solution**:
   - The presenters will create a video to present just the part of the solution they worked on. It will take too long to present the entire solution.

Payment Module



[Logic 1] Will do a flowchart of the hardware states (both pseudocode and flowchart groups).
[Logic 2] Will handle the software logic for selecting products for purchase.
[Logic 3] Will do software logic for paying for purchase and adjusting inventory.

Task	Subtask	Member(s)	Marks	Comments
Pseudocode	Logic 1	4	40%	
	Logic 2	5	40%	
	Logic 3	6	40%	
	Combined	4-6	60%	
FlowChart	Logic 1	1	40%	
	Logic 2	2	40%	
	Logic 3	3	40%	
	Combined	1-3	60%	
Video	Presentation	1 or 4	100%	Members rotate weekly