Name: Luca Nasrallah
ID: 40316548
Section: COMP 248 Section R – Fall 2024

Assignment 3 – Algorithms for Questions 1 and 2

Question 1: **Algorithm for ATM Program*

1. **Start Program:**

- Print the welcome message: "Welcome to Concordia Virtual ATM service."

2. **Declare Variables:**

- Create a `Scanner` object (`cin`) to accept user input.
- Define `int` variable `displayMenuChoice` to store the user's choice in the main menu, initialized to 0.
- Define `double` variable `storedBalance` to keep track of the account balance, initialized to 0.
- Define constants:
  - `final double `MAX_DEPOSIT` = 100000 for the maximum deposit limit.
  - `final double `MIN_VALUE` = 5 for the minimum transaction amount.
  - `final double MAX_WITHDRAWAL = 1000` for the maximum withdrawal limit.
- Define `double` variable `inputAmount` for handling deposit and withdrawal amounts.
- Define boolean variable `exit` initialized to `false` to control loops for deposit/withdrawal operations.

3. **Main Program Loop:**

- Repeat until the user chooses to exit (`displayMenuChoice != 4``):

  Print the menu

  1. Check Balance
  2. Deposit
  3. Withdraw
  4. Exit

  - Prompt the user to enter their choice and store it in `displayMenuChoice`.

4. **Handle User Choice:**

- If `displayMenuChoice` is between 1 and 3:
  - Use a switch statement to handle each specific choice:

    ****Case 1 - Check Balance**:**

```
- Print the current balance, formatted to two decimal places (e.g., "Current balance is: storedBalance$").


**Case 2 - Deposit**:

- Repeat the following steps until a valid deposit is made (`exit = true`):
  - Prompt the user to enter deposit amount and store it in inputAmount.
  - **Check validity**:
    - If inputAmount` < MIN_VALUE or inputAmount > MAX_DEPOSIT:
      - Print "The Input amount is INVALID, not between 5.00$ and 10,000.00$".
      - Set exit to `false`.
    - Else if  inputAmount is not a multiple of 5:
      - Print "The Input amount is INVALID, was not multiple of 5, 10, 50, or 100".
      - Set exit to `false`.
    - Else:
      - Add inputAmount to storedBalance.
      - Print "Deposit Successful!".
      - Set exit to true.


**Case 3 - Withdraw**:

- Repeat the following steps until a valid withdrawal is made (exit = true):
    - Prompt the user to enter withdrawal amount (only in multiples of 5, 10, 50, or 100) and store it in
inputAmount.
    - Calculate the new balance after withdrawal: withdrawalBalance = storedBalance -
inputAmount.
  - **Check validity**:
    - If inputAmount < MIN_VALUE or inputAmount > MAX_WITHDRAWAL:
      - Print "The withdrawal amount is INVALID, not between 5.00$ and 1,000$".
      - Set exit to false.
    - Else if inputAmount is not a multiple of 5:
      - Print "The withdrawal amount is INVALID, was not multiple of 5, 10, 50, or 100".
      - Set exit to false.
    - Else if withdrawalBalance is negative:
```

~~~
       - Print "There is insufficient funds inside your account to withdraw such amount. Please try again."
       - Set exit to `false`.
      - Else:
       - Print "You will receive:", and calculate number of bills:
        - fiftyBills = inputAmount / 50
        - Update inputAmount to remainder: inputAmount %= 50
        - tenBills = inputAmount / 10
        - Update inputAmount to remainder: inputAmount %= 10
        - fiveBills = inputAmount / 5
       - Print the number of bills:
        ```
        Number of 50.00$ bills: [fiftyBills]
        Number of 10.00$ bills: [tenBills]
        Number of 5.00$ bills: [fiveBills]
        ```
       - Print "Withdrawal Successful!" and update storedBalance to withdrawalBalance.
       - Set exit to true.
  - **Else if displayMenuChoice > 4:
   - Print "The Input for the display Menu is INVALID, please try again."
5. **Exit Program**:
  - Print "Thank you for using the Concordia Virtual ATM Service. Have a great day! :)".
6. **End Program**.
~~~

Question 2: Algorithm for Product Management System

````
1. **Start Program**:
````

- Print the welcome message: "Welcome to the Product Management System Program. Let's begin".

2. **Declare Variables**:

  - Create a `Scanner` object cin for reading user input.

  - Define int variables:

    - displayMenu initialized to 0 for storing main menu choices.

    - choice initialized to 0 for storing user menu selections.

    - final int ARRAY_SIZE = 5 for the fixed size of arrays (five products).

  - Define `boolean` variable exit initialized to `false` for loop control.

  - Define `String` variable searchWord to store product names for search queries.

  - Define three arrays:

    - String[] products = new String[ARRAY_SIZE] to store product names.

    - double[] prices = new double[ARRAY_SIZE] to store product prices.

    - int[] quantity = new int[ARRAY_SIZE] to store product quantities.

3. **Collect Product Information**:

  - Prompt the user to enter details for 5 products: each entry includes price, quantity, and name.

  - For each product (from 1 to 5):

    - Prompt the user to enter:

      - Price as a `double` stored in prices[i]

      - Quantity as an `int` stored in quantity[i]

      - Product name as a `String` stored in products[i]

4. **Main Program Loop**:

  - Repeat until the user chooses to exit (exit = true):

    - Display the menu options:

      ```

      1. Display Information of all Products

      2. Update the quantity of a product

      3. Search for a product by name

      4. Find the product with the lowest quantity
      ```

5. Find the product with the highest price

6. Exit

` ` `

- Prompt the user to input their choice and store it in choice.

5. **Handle User Choice**:

- If choice is out of bounds (not between 1 and 6):
  - Print "This is an invalid input, please try again."
  - Set exit to false.
- If choice is 6:
  - Set exit to true to break out of the main loop
- Otherwise, proceed with switch cases for choices 1 through 5:

Case 1 - Display Information of All Products:

- Print "Product List" and for each product:
  - Display Name, Price (formatted to 2 decimals), and Quantity.
- Set exit to `false`.

Case 2 - Update the Quantity of a Product:

- Repeat until valid input is provided:
  - Prompt the user to enter a product number (1 to 5), store it in productNumber.
  - **If productNumber is valid (1 to 5):
    - Prompt the user to enter new quantity for the selected product.
    - **If the new quantity is non-negative**:
      - Update quantity[productNumber - 1]to the new quantity.
      - Print "Quantity updated successfully!".
      - Set exit to `true`.
    - Else, print an error message: "The value entered is INVALID, please try again."
  - Else, print "The input is INVALID, please try again." (For an invalid initial statement.)
- Set exit to `false`.

```
**Case 3 - Search for a Product by Name**:
- Set boolean found = false
- Repeat until a product is found:
  - Prompt the user to enter a product name and store it in searchWord.
  - For each product in products:
    - If searchWord matches a product name (ignoring case):
      - Display that product's Name, Price, and Quantity.
      - Set exit to `true` and found = true. To end the loop
  - If no match (found = false):
    - Print "Product not found. Please try again."
    - Set exit to `false`.
- Set exit to `false`. To restart the loop.

**Case 4 - Find the Product with the Lowest Quantity**:
- Initialize `int` min to 0 (index of product with the lowest quantity).
- For each product:
  - If quantity[i] < quantity[min], update min to i.
- Print the product with the lowest quantity, displaying Name, Price, and Quantity.
- Set exit to false. For the main loop.

**Case 5 - Find the Product with the Highest Price**:
- Initialize `int` max to 0 (index of product with the highest price).
- For each product:
  - If prices[i]> prices[max], update max to i.
- Print the product with the highest price, displaying Name, Price, and Quantity.
- Set exit to `false`.
```

6. **Exit Program**:

 - Print the closing message: "Thank you for using the most amazing product Inventory Management program that you have ever seen. Have a great day!

7. **End Program**:

   - Close the `Scanner` `cin`