A retail banking system where customers perform various transactions, such as depositing money, withdrawing funds, or transferring funds between accounts. You are tasked with developing the C# application for the bank.

**Functional Requirement:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Req. #** | **Requirements Description** | **Class Name** | **Method Name** | **Parameters** | **Description** |
| **1** | Add transaction details to the queue in the program class. | **Transaction** | AddTransactionToQueue | Transaction transaction | This method **adds**the transactions to the **transactionQueue**property, which is implemented as Queue. (already given in Program class) |
| **2** | This method handles transactions sequentially, ensuring they are processed in the order they were received. | **Transaction** | ProcessTransactions | - | This method returns the string value.  **Constraints:**   * This method should retrieve the transactions in first in first out order from **transactionQueue**. * When the transaction amount is less than zero, return **"Enter the valid amount"** * When the amount is greater than zero return **"Transaction <Transaction Number> processed. Account Holder: <Account Holder Name>, Amount: <Transaction Amount>"** |

**Sample Input/Output 1:**

Enter the number of transactions

**3**

Enter the details (transaction number, name, amount)

**111**

**John**

**1000**

Transaction 111 processed. Account Holder: John, Amount: 1000

Enter the details (transaction number, name, amount)

**222**

**Lary**

**500**

Transaction 222 processed. Account Holder: Lary, Amount: 500

Enter the details (transaction number, name, amount)

**333**

**David**

**100**

Transaction 333 processed. Account Holder: David, Amount: 100

All transactions processed.

**Sample input/Output 2:**

Enter the number of transactions

**1**

Enter the details (transaction number, name, amount)

**111**

**John**

**-11000**

Enter the valid amount

2. A professional writer is using a text editor to compose an article. They type a paragraph and decide to delete the last sentence. Later, they realize they want to revert the changes and undo the deletion.You are tasked with developing the C# application for them to make their work easy.

**Functional Requirement:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Req. #** | **Requirements Description** | **Class Name** | **Method Name** | **Parameters** | **Description** |
| **1** | Delete the last word if needed | **TextEditor** | DeleteLastCharacter | string text | This method Adds the text given by the user to **undoStack**which is implemented as a stack property in the program class(already provided) and deletes the last character of the sentence entered by the user when they need to delete.  **Constraints:**   * This method returns the result in string format as **<result>\_** |
| **2** | Undo the deletion if needed | **TextEditor** | UndoDeletion | - | This method undo the deletion of the last character when they need to undo and return the string.  **Constraints:**   * Retrieve the previously deleted text from the stack property in the Program class. * Set the retrieved value to the **text**variable. * Return the modified text. |

**Sample Input/Output 1:**

Enter the text

**Hello Everyone!!! I am Robot**

Enter yes, if you need to delete the last letter in your sentence

**yes**

Your sentence after deletion of last letter is :

Hello Everyone!!! I am Robo\_

Enter yes, if you need to undo the last deleted letter

**yes**

Your sentence after undoing the deletion is :

Hello Everyone!!! I am Robot

**Sample Input/output 2:**

Enter the text

**Hello Everyone!!! I am Robot**

Enter yes, if you need to delete the last letter in your sentence

**No**

Your editing completed

3. DigiVerse, a reputable numerical exploration platform, is set to revolutionize data analysis with the development of a C# program. The Numeric Harmony Analyzer within DigiVerse enables users to input numeric sequences, applying criteria to identify even numbers for squaring and odd numbers for cube operations, unveiling hidden patterns in the digital landscape.

**Constraints:**

* The sequence should be numbers and space-separated; otherwise, print "**Invalid sequence**".

**Note:**

* Do not edit the existing code template.
* In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user, and the rest of the text represents the output.
* Implement the business requirements within the main method. Please do not change the class name.
* Do not use **Environment.Exit()** to terminate the program.

**Sample Input / Output 1**

Enter the numeric sequence (space-separated):

**2 5 8 12 17**

4 125 64 144 4913

**Sample Input / Output 2**

Enter the numeric sequence (space-separated):

**1 2 3 a**

Invalid sequence

-------------------------

4. In the heart of an industrial district, Modern Textiles Inc. stands as a pioneering manufacturer of high-quality synthetic fibers. As a software consultant specializing in process optimization, you've been tasked with developing a C# application to streamline the synthesis of nylon, ensuring efficiency and precision in the manufacturing process.

**Functional Requirements:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Req. #** | **Requirements Description** | **Class Name** | **Method Name** | **Parameters** | **Description** |
| **1** | Validate the synthesis parameters entered by the user | **SynthesisUtility** | ValidateSynthesis | - | This method validates the details entered by the user.  The valid specifications include:   |  |  | | --- | --- | | **Temperature** | Between 1 and 300 (inclusive) | | **Pressure** | Between 1 and 10 (inclusive) | | **Reaction time** | Greater than 0 |   **Constraint:**   * This method returns **true**when the specifications are valid; otherwise, it returns **false** |
| 2 | Execute the synthesis process and provide feedback to the user | **SynthesisUtility** | PerformSynthesis | - | This method returns the int value.  **Constraints:**   * Synthesis calculation is performed by multiplying the temperature and pressure * Divide the above-resulting value by the reaction time. |

**SynthesisUtility**classneeds to refer the class.

**Sample Input 1:**

Enter the Primary Monomer:

**Adipoyl chloride**

Enter the Secondary Monomer:

**Hexamethylene diamine**

Enter the Reaction Temperature (in Celsius):

**250**

Enter the Reaction Pressure (in atm):

**5**

Enter the Catalyst:

**Sulfuric Acid**

Enter the Reaction Duration (in minutes):

**60**

**Sample Output 1:**

Performing synthesis of Nylon using Adipoyl chloride and Hexamethylene diamine...

Applying 250 degree Celsius and 5 atm pressure with Sulfuric Acid catalyst...

Reacting for 60 minutes...

The nylon synthesis results 20 as calculated value...

Nylon synthesis completed successfully!

**Sample Input 2:**

Enter the Primary Monomer:

**Adipoyl chloride**

Enter the Secondary Monomer:

**Hexamethylene diamine**

Enter the Reaction Temperature (in Celsius):

**350**

Enter the Reaction Pressure (in atm):

**12**

Enter the Catalyst:

**Sulfuric Acid**

Enter the Reaction Duration (in minutes):

**0**

**Sample Output 2:**

Enter the valid specifications

5. A prestigious film studio is embarking on its next major project: a trilogy of films spanning various genres. To streamline the production process and accurately estimate costs, they've sought the assistance of a software developer. Your objective is to develop a C# application that validates movie specifications and computes the production cost for each movie in the trilogy.

**Functional Requirements:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Req. #** | **Requirements Description** | **Class Name** | **Method Name** | **Parameters** | **Description** |
| **1** | Validate the movie specifications entered by the user | **MovieUtility** | ValidateMovieSpecification | string genre, double mainActorSalary | This method is used to **validate**the movie specifications.  **Valid**Movie Specifications:   * Genre should be "**Action**", "**Drama**" or "". * The Main Actor's salary should be **greater**than **0**.   **Constraints:**   * When the specifications are valid, return **true;** Otherwise, return **false**. * The movie genre is **case-sensitive**. |
| **2** | Calculate the production cost of the movie using genre, main actor's salary. | **MovieUtility** | CalculateProductionCost | string genre, double mainActorSalary | This method calculates the production cost and returns double value.   |  |  |  | | --- | --- | --- | | **Genre** | **Cost (per genre)** | **Actor Salary** | | Action | 500000 | Three times of main actor's salary | | Drama | 300000 | N/A | | Comedy | 200000 | N/A |     **Constraints:**   * The production cost is calculated by adding the genre cost and the actor's salary(for available genre). |

**Sample Input 1:**

Enter Movie Genre (Action/Drama/Comedy):

**Action**

Enter Main Actor's Salary:

**100000**

**Sample Output 1:**

Production Cost: 800000

**Sample Input 2:**

Enter Movie Genre (Action/Drama/Comedy):

**drama**

Enter Main Actor's Salary:

**100000**

**Sample Output 2:**

Invalid movie specifications

6. Jessica is a manager at a retail store and wants to create a program to track the sales performance of her employees. Write a C# program to assist Jessica in finding the first day when an employee meets or exceeds a certain sales target.

**For example**, consider a scenario where an employee's daily sales for a week are $100, $120, $90, $150, $110, $130, and $140 respectively. Jessica wants to determine the first day when the employee's sales reach or exceed $150. In this scenario, the first qualifying day is determined to be Day 4 because the employee's sales reach $150 on that day.

**Constraints:**

* When the employee meets or exceeds the sales target on any day, display "**Congratulations! The employee met or exceeded the sales target of $<targetAmount> on the <qualifyingDay>-day**".
* When no day meets or exceeds the sales target, display "**No qualifying day found".**

**Note:**

* Do not edit the existing code template.
* In the Sample Input / Output provided, the highlighted text in bold corresponds to the input given by the user, and the rest of the text represents the output.
* Implement the business requirements within the main method. Please do not change the class name.
* Do not use **Environment.Exit()** to terminate the program.

**Sample Input / Output 1**

Enter the number of days

**6**

Enter the sales amount for Day 1

**120**

Enter the sales amount for Day 2

**130**

Enter the sales amount for Day 3

**170**

Enter the sales amount for Day 4

**140**

Enter the sales amount for Day 5

**- 3**

Enter the sales amount for Day 6

**200**

Enter the sales target

**160**

Congratulations! The employee met or exceeded the sales target of $160 on the 3-day

**Sample Input / Output 2**

Enter the number of days

**4**

Enter the sales amount for Day 1

**50**

Enter the sales amount for Day 2

**80**

Enter the sales amount for Day 3

**90**

Enter the sales amount for Day 4

**40**

Enter the sales target

**100**

No qualifying day found