Debugging a Task Manager Program: Identifying and Fixing Logical Errors

Objective:

In this activity, you will debug a pre-existing task manager program. They will identify and fix logical errors in the task addition and completion functions, ensuring tasks are stored correctly, marked as completed, and displayed accurately.

Steps

Step-by-Step Code Construction:

1. **Identify Errors in List Usage**: The code uses lists to store tasks and their completion status. If you're unfamiliar with lists, consider them collections that can hold multiple values (similar to arrays). Check if the tasks and their statuses are being accessed correctly.

- Review how tasks are added to the list.
- Ensure that you can loop through the list and display tasks properly using a loop.
- Debug any issues related to list indexing or adding tasks to the list.

2. **Validate User Input for Task Numbers**: The program asks users to input numbers to select tasks. You need to ensure that the input is correctly validated and parsed.

- Use **int.TryParse** to make sure the user input is a valid number.
- Check that the code correctly handles invalid input (non-numeric or out-of-range values).
- Ensure that tasks are accessed by valid numbers (between 1 and the number of tasks available).

***Hint*: You can guide users with a friendly error message when input is incorrect.**

3. **Debug the Menu and Control Flow**: The program uses a switch statement to navigate the main menu. Make sure that each option works correctly:

- Check that each menu option (e.g., view tasks, mark tasks complete) correctly leads to the desired functionality.
- Ensure that the program returns to the main menu after each action is completed.

***Hint*: You should also confirm that the "Exit" option exits the program properly.**

4. **Check Loop Logic for Viewing and Marking Tasks**: The code contains loops for viewing tasks and marking them as complete. Debug any issues with how the loops process each task.

- Ensure the loop correctly iterates through each task in the list when displaying them.
- Verify that marking a task as complete updates the task's status accurately.

5. **Test for Boundary Cases**: The program should handle different cases, including:

- What happens when there are no tasks?
- What if the user tries to mark a task that doesn't exist?

6. **Ensure the code properly handles these scenarios without crashing or producing incorrect results.**

7. **Final Testing**: After debugging, run the program multiple times to test its functionality. In order to run the code, visit Dot Net Fiddle, delete the code on the page, paste your code into the left-hand side, and select run.

- Add several tasks and view them.
- Mark tasks as complete and check if the status updates correctly.
- Try invalid inputs (non-numeric, negative numbers, numbers too large) and ensure the program handles them gracefully.

Code:

```
using System;
using System.Collections.Generic;

class TaskManager
{
    static List<string> tasks = new List<string>();
    static List<bool> taskStatus = new List<bool>();

    static void Main(string[] args)
    {
        while (true)
        {
            Console.WriteLine("Task Manager");
            Console.WriteLine("1. Add Task");
            Console.WriteLine("2. Mark Task as Completed");
            Console.WriteLine("3. View Tasks");
            Console.WriteLine("4. Exit");
Console.WriteLine("What would you like to do? (choose 1-4)");

            string choice = Console.ReadLine();
```

```
            switch (choice)
            {
                case "1":
                    AddTask();
                    break;
                case "2":
                    CompleteTask();
                    break;
                case "3":
                    ViewTasks();
                    break;
                case "4":
                    return;
                default:
                    Console.WriteLine("Invalid choice, try again.");
                    break;
            }
        }
    }

    static void AddTask()
    {
        Console.WriteLine("Enter task description:");
        string task = Console.ReadLine();
        tasks.Add(task);
        taskStatus.Add(false);  // Marking as not completed by default
        Console.WriteLine("Task added successfully.");
    }

    static void CompleteTask()
    {
        Console.WriteLine("Enter task number to mark as completed:");
        int taskNumber = int.Parse(Console.ReadLine());

        if (taskNumber < 0 || taskNumber >= tasks.Count)
        {
            Console.WriteLine("Invalid task number.");
            return;
        }

        taskStatus[taskNumber] = true;
        Console.WriteLine($"Task '{tasks[taskNumber]}' marked as
completed.");
    }

    static void ViewTasks()
    {
        Console.WriteLine("Tasks:");
        for (int i = 0; i <= tasks.Count; i++)
        {
            string status = taskStatus[i] ? "Completed" : "Pending";
            Console.WriteLine($"{i + 1}. {tasks[i]} - {status}");
        }
    }
}
```