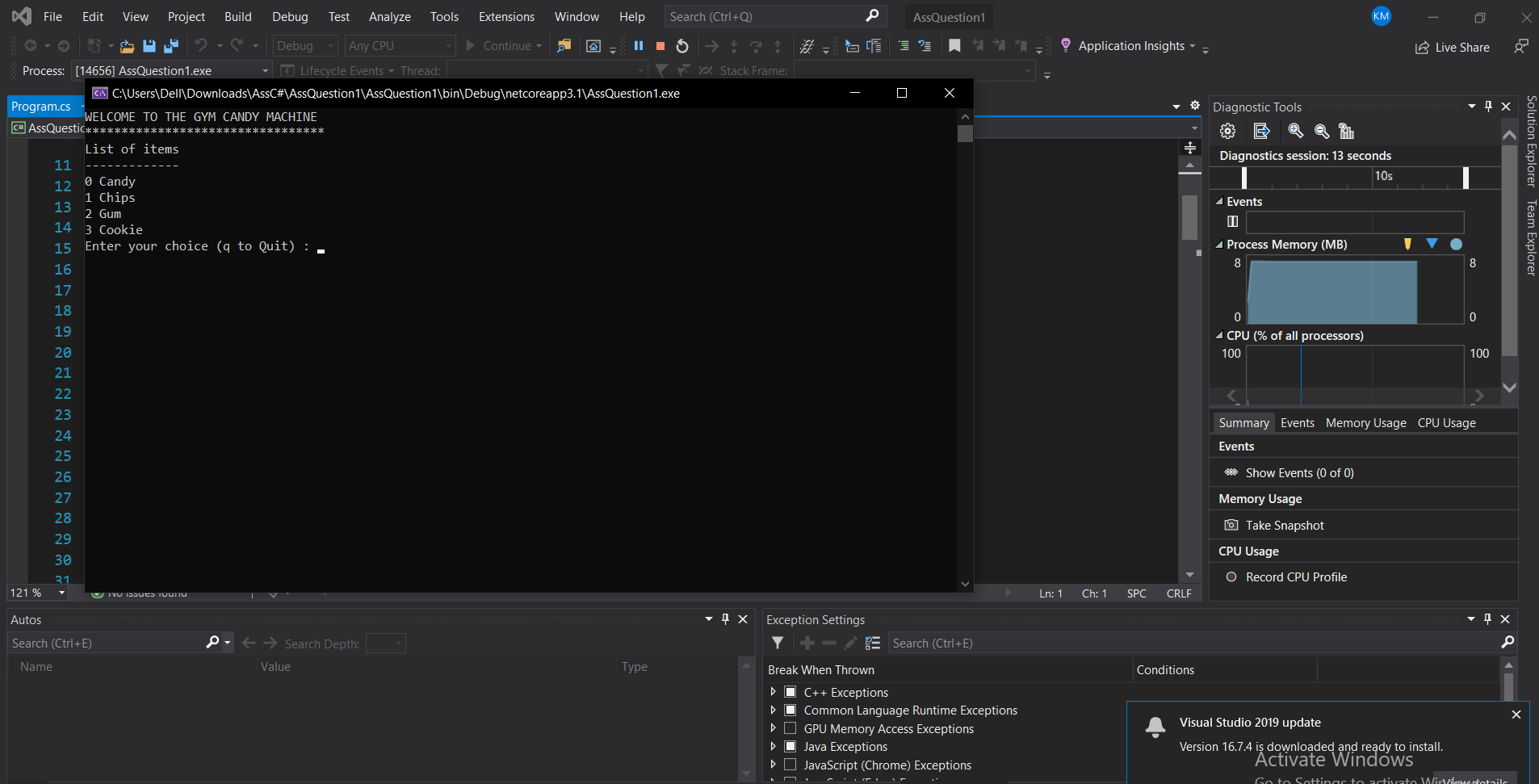
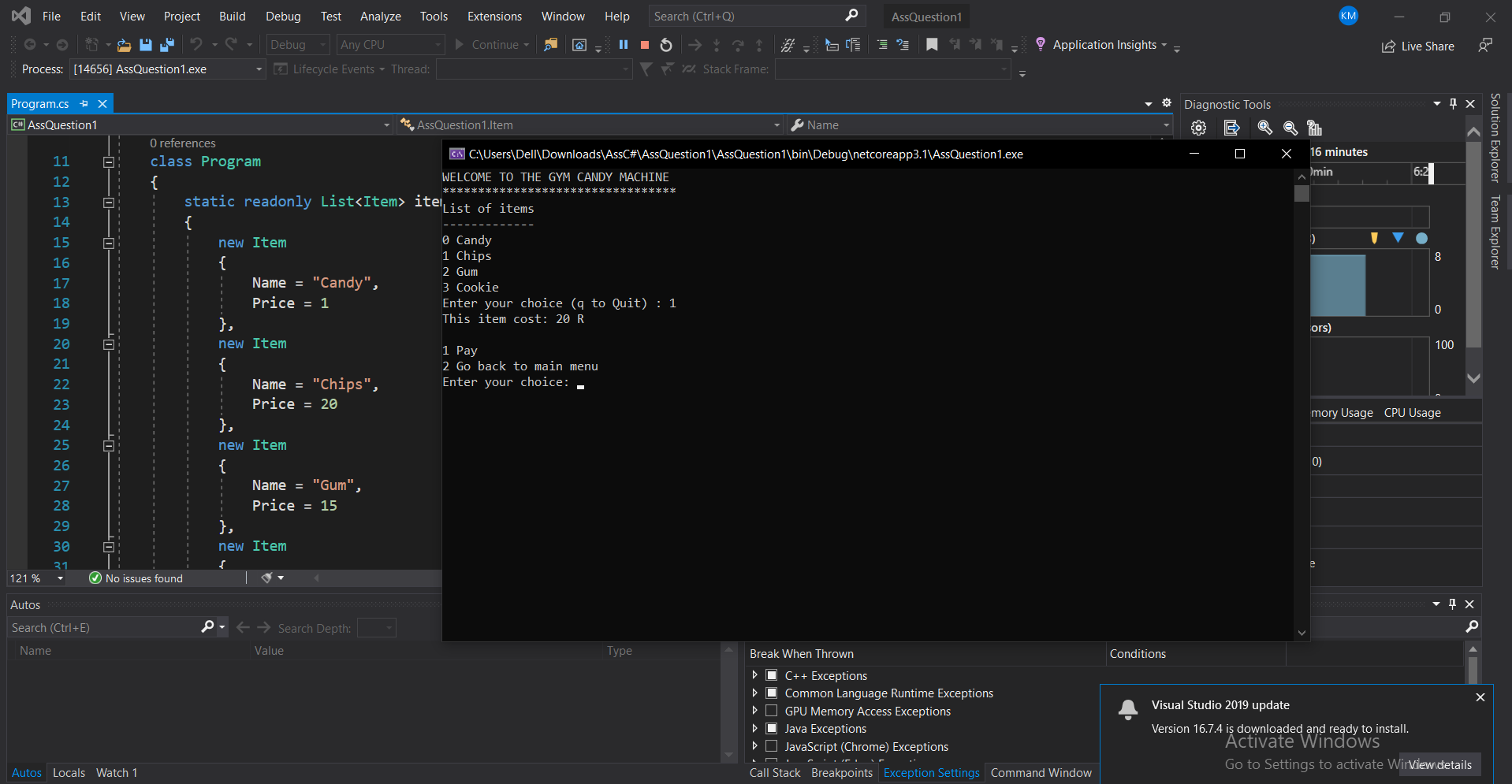
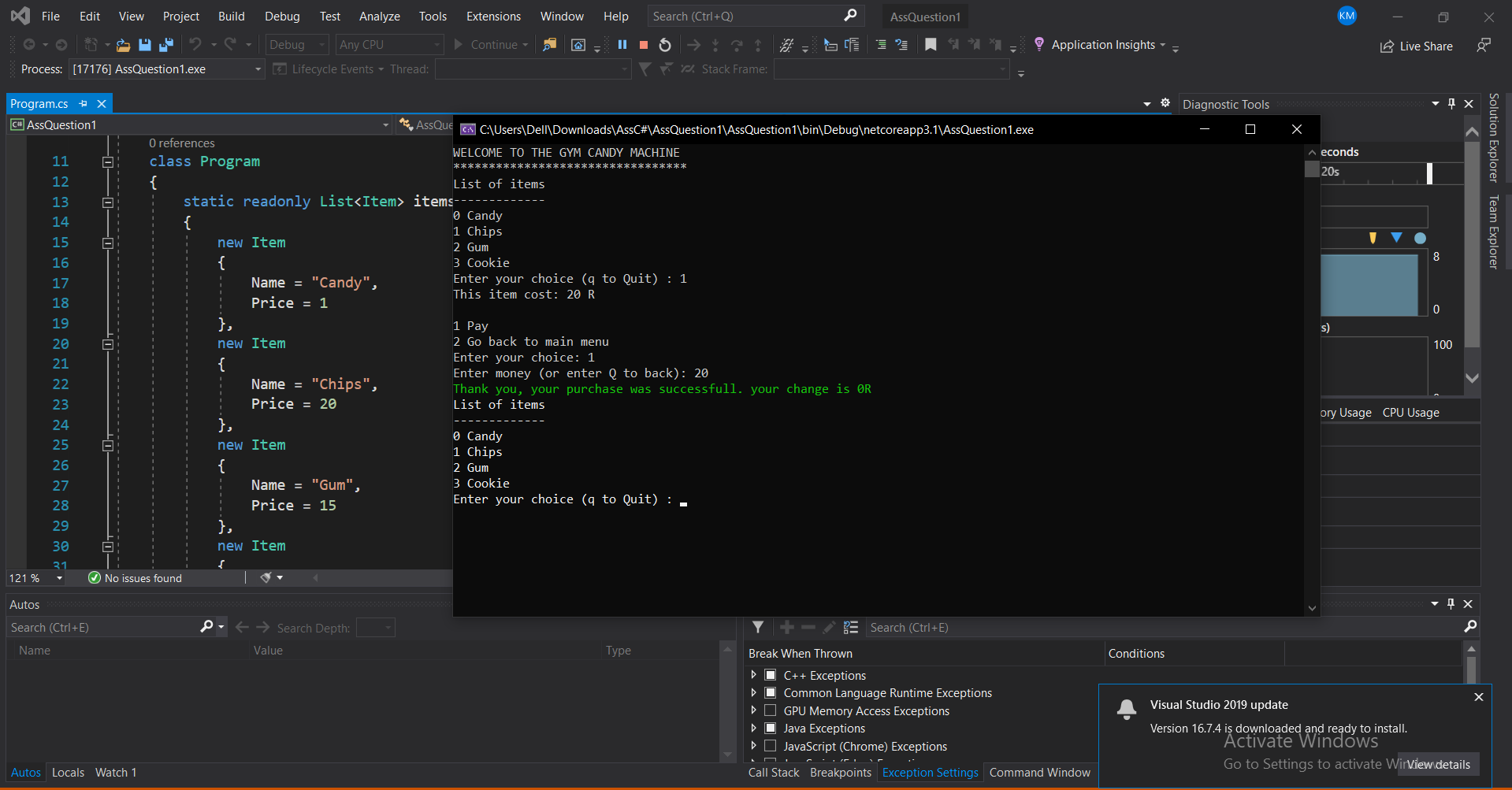
|  |
| --- |
| using System;  using System.Collections.Generic;  namespace AssQuestion1  {  class Item  {  public string Name { get; set; }  public decimal Price { get; set; }  }  class Program  {  static readonly List<Item> items = new List<Item>  {  new Item  {  Name = "Candy",  Price = 1  },  new Item  {  Name = "Chips",  Price = 20  },  new Item  {  Name = "Gum",  Price = 15  },  new Item  {  Name = "Cookie",  Price = 5  }  };  static void Main(string[] args)  {  Console.WriteLine("WELCOME TO THE GYM CANDY MACHINE");  Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  while(true)  {  Console.WriteLine("List of items");  Console.WriteLine("-------------");  PrintItems();  Console.Write("Enter your choice (q to Quit) : ");  var input = Console.ReadLine();  if (input.ToLower() == "q")  {  break;  }  else  {  SelectItem(input);  }  }  }  static void PrintItems()  {  for(var i = 0; i < items.Count; i++)  {  var item = items[i];  Console.WriteLine($"{i} {item.Name}");  }  }  static void SelectItem(string itemSelection)  {  if (int.TryParse(itemSelection, out var itemIndex))  {  if (itemIndex < 0 || itemIndex >= items.Count)  {  PrintError($"Invalid section. your selection must range from 0 to {items.Count - 1}");  }  else  {  var done = false;  while(!done)  {  var item = items[itemIndex];  Console.WriteLine($"This item cost: {item.Price} R");  Console.WriteLine("\n1 Pay\n2 Go back to main menu");  Console.Write("Enter your choice: ");  var selectionInput = Console.ReadLine();  if (int.TryParse(selectionInput, out var selection))  {  switch(selection)  {  case 1:  BuyItem(item);  done = true;  break;  case 2:  done = true;  break;  default:  PrintError("Invalid selection your selection must be either 0 or 1");  break;  }  }  else  {  PrintError("Invalid selection your selection must be a valid number");  }  }  }  }  else  {  PrintError("Invalid selection, your selection must be a number");  }  }  static void BuyItem(Item item)  {  while(true)  {  Console.Write("Enter money (or enter Q to back): ");  var moneyInput = Console.ReadLine();  if (moneyInput.ToLower() == "q") break;  if (int.TryParse(moneyInput, out var money))  {  if (money < item.Price)  {  PrintError("Unsufficient fund.");  }  else  {  PrintSuccess($"Thank you, your purchase was successfull. your change is {money - item.Price}R");  break;  }  }  else  {  PrintError("Your money must be a valid number");  }  }  }    static void PrintError(string message)  {  Console.ForegroundColor = ConsoleColor.Red;  Console.WriteLine(message);  Console.ForegroundColor = ConsoleColor.White;  }  static void PrintSuccess(string message)  {  Console.ForegroundColor = ConsoleColor.Green;  Console.WriteLine(message);  Console.ForegroundColor = ConsoleColor.White;  }  }  } |

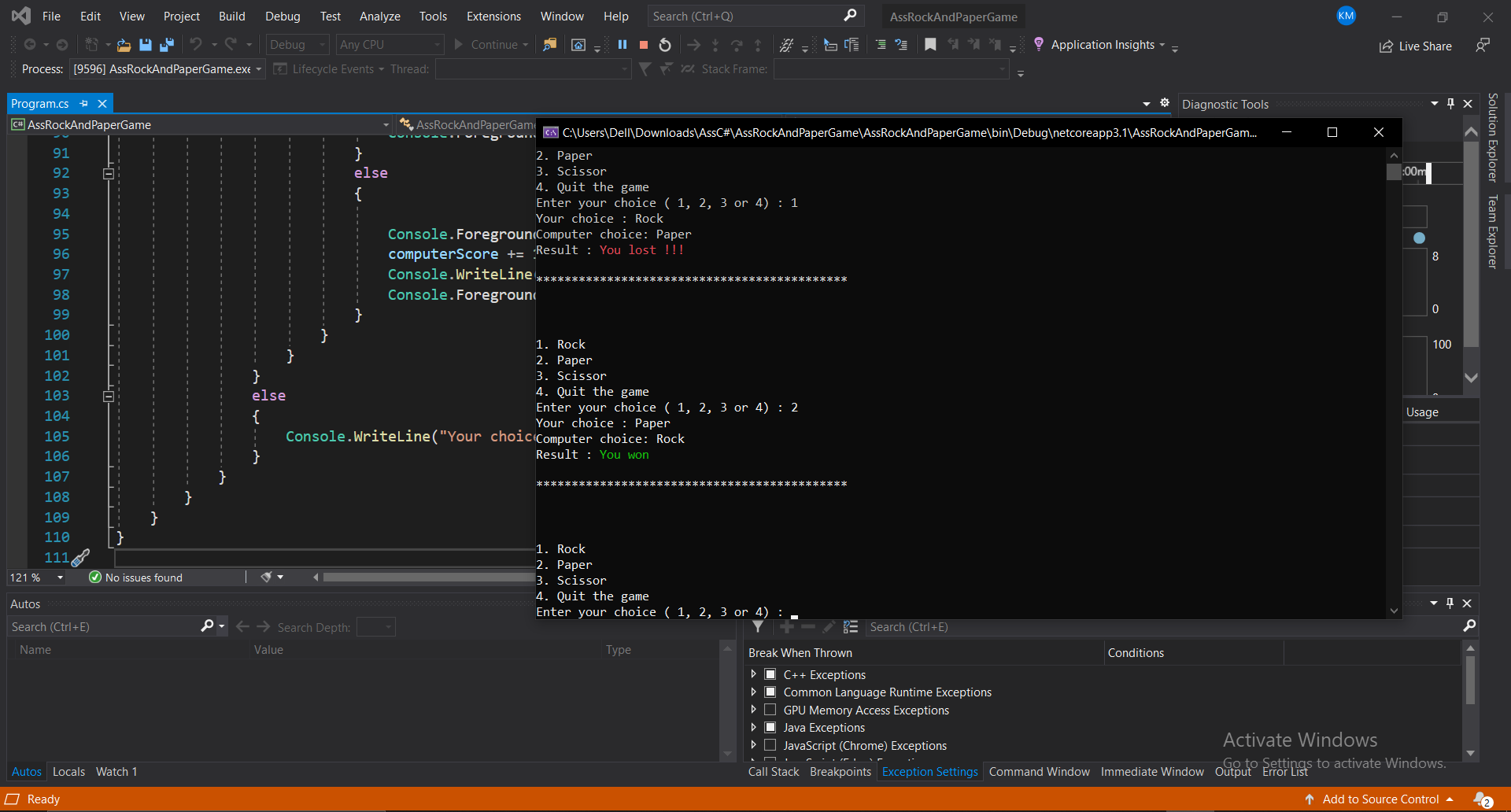


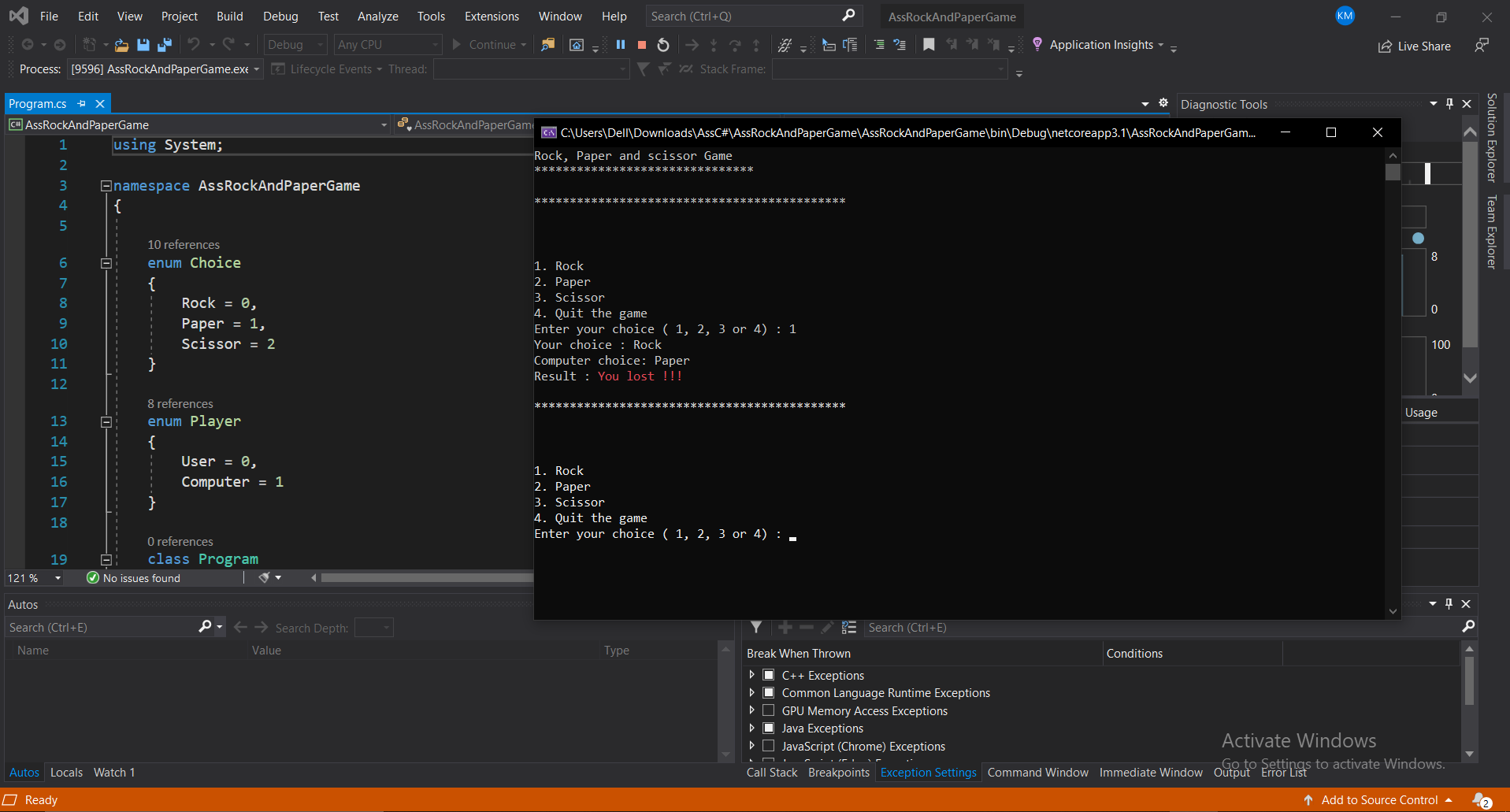




2. QUESTION 2

|  |
| --- |
| using System;  namespace AssRockAndPaperGame  {  enum Choice  {  Rock = 0,  Paper = 1,  Scissor = 2  }  enum Player  {  User = 0,  Computer = 1  }  class Program  {  static readonly Choice[] Choices = new[] { Choice.Rock, Choice.Paper, Choice.Scissor };  static readonly Random rnd = new Random();  static void Main(string[] args)  {  Console.WriteLine("Rock, Paper and scissor Game");  Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  int computerScore = 0;  int userScore = 0;  while (true)  {  Console.WriteLine("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  Console.WriteLine("\n\n");  Console.WriteLine("1. Rock\n2. Paper \n3. Scissor \n4. Quit the game");  Console.Write("Enter your choice ( 1, 2, 3 or 4) : ");  var choiceInput = Console.ReadLine();  var tryGetChoice = int.TryParse(choiceInput, out var choice);  if (tryGetChoice)  {  if (choice < 1 || choice > 4)  {  Console.WriteLine("That was an invalid choice");  }  else  {  if (choice == 4)  {  // Quit the game  Console.WriteLine("You have quit the game");  Console.WriteLine($"Computer : {computerScore}");  Console.WriteLine($"You : {userScore}");  Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");  var result = userScore == computerScore ? "Draw" : userScore > computerScore ? "You won" : "You lost :)";  Console.WriteLine($"Final Result : {result}");  break;  }  else  {  var computerChoice = Choices[rnd.Next(0, 2)];  var userChoice = Choices[choice - 1];  Console.WriteLine($"Your choice : {userChoice}");  Console.WriteLine($"Computer choice: {computerChoice}");  Console.Write("Result : ");  if (computerChoice == userChoice)  {  Console.Write("It's a Tie");  continue;  }  Player? winner = null;  switch (userChoice)  {  case Choice.Rock:  winner = computerChoice == Choice.Scissor ? Player.User : Player.Computer;  break;  case Choice.Paper:  winner = computerChoice == Choice.Rock ? Player.User : Player.Computer;  break;  case Choice.Scissor:  winner = computerChoice == Choice.Paper ? Player.User : Player.Computer;  break;  }  if (winner == Player.User)  {  Console.ForegroundColor = ConsoleColor.Green;  userScore += 1;  Console.WriteLine("You won");  Console.ForegroundColor = ConsoleColor.White;  }  else  {    Console.ForegroundColor = ConsoleColor.Red;  computerScore += 1;  Console.WriteLine("You lost !!!");  Console.ForegroundColor = ConsoleColor.White;  }  }  }  }  else  {  Console.WriteLine("Your choice must be a number raging from 1 to 4");  }  }  }  }  } |





QUESTION 3

|  |
| --- |
| using System;  using System.Collections.Generic;  namespace Question3  {  public class Program  {  private static Board currentBoard;  private static Player currentPlayer;  private static List<string> tempChain;  public static void Main(string[] args)  {  Console.WriteLine("Welcome to TicTacToe Autism Edition v0.0001");  currentBoard = new Board();  tempChain = new List<string>(3);  NewGame();  }  private static void NewGame()  {  currentBoard.NewGame();  currentPlayer = Shuffle();  Console.WriteLine($"Board size is {currentBoard.Size}. Player {currentPlayer.Name} has the first turn:");  drawCurrentBoard();  processGame();  }  private static void processGame()  {  if (currentBoard.TurnCount >= 9)  {  endGameDraw();  }  Console.WriteLine($"{currentPlayer.Name}, enter the number from 1 to 9, each number indicates square on the board: ");  int fieldNumber = processInput(Console.ReadLine());  processMove(currentPlayer, fieldNumber);  }  private static void processMove(Player curPlayer, int move)  {  if (currentBoard.CurrentState[move - 1] == " ")  {  Console.WriteLine($"{currentPlayer.Name} takes square {move}");  currentBoard.CurrentState[move - 1] = currentPlayer.Char;  currentBoard.TurnCount++;  currentPlayer.Turns++;  currentPlayer.movesChain.Add(move);  drawCurrentBoard();  if (checkWin() == false)  {  currentPlayer = nextPlayer();  processGame();  }  else  {  endGameWin();  }  }  else  {  Console.WriteLine($"Board square {move} is already occupied by \"{currentBoard.CurrentState[move - 1]}\"");  processGame();  }  }  private static int processInput(string input)  {  int number = -1;  while (number == -1)  {  try  {  number = Int32.Parse(input);  if (number >= 1 && number <= 9)  { return number; }  else  {  Console.WriteLine($"{input} is not a valid number in 1-9 range, try again: ");  number = -1;  input = Console.ReadLine();  }  }  catch (Exception)  {  Console.WriteLine($"{input} is not a valid number in 1-9 range, try again: ");  number = -1;  input = Console.ReadLine();  }  }  return number;  }  private static void processGameMode(string input)  {  while (input.ToLower() != "new" || input.ToLower() != "rematch")  {  if (input.ToLower() == "new")  {  NewGame();  break;  }  else if (input.ToLower() == "rematch")  {  Rematch();  break;  }  else  {  Console.WriteLine("Unknown command, try again!");  input = Console.ReadLine();  continue;  }  }  }  private static void drawCurrentBoard()  {  var z = currentBoard.CurrentState;  Console.Write($" {z[6]} | {z[7]} | {z[8]} \n");  Console.Write($"---+---+---\n");  Console.Write($" {z[3]} | {z[4]} | {z[5]} \n");  Console.Write($"---+---+---\n");  Console.Write($" {z[0]} | {z[1]} | {z[2]} \n");  }  private static Player nextPlayer()  {  if (currentPlayer == currentBoard.Player1) { return currentBoard.Player2; }  else { return currentBoard.Player1; }  }  private static Player Shuffle()  {  Random randomID = new Random();  int rpl = randomID.Next(1, 101);  if (rpl % 2 == 0) { return currentBoard.Player1; }  else { return currentBoard.Player2; }  }  private static bool checkWin()  {  if (currentPlayer.Turns < 3 || currentPlayer.movesChain.Count < 3)  { return false; }  else  {  tempChain.Clear();  for (int i = 1; i <= 3; i++)  {  tempChain.Add(currentPlayer.movesChain[currentPlayer.movesChain.Count - i].ToString());  }  tempChain.Sort();  var chain = string.Join<string>("", tempChain);  if (chain == "147" || chain == "258" || chain == "369" || chain == "789" || chain == "456" || chain == "123" || chain == "357" || chain == "159")  { return true; }  else { return false; }  }  }  private static void endGameWin()  {  currentBoard.TotalGames++;  currentPlayer.Wins++;  Console.WriteLine($"Congratulations, player {currentPlayer.Name} won!");  Console.WriteLine($"Game stats: turns this round - {currentBoard.TurnCount} | total rounds - {currentBoard.TotalGames}");  Console.WriteLine($"Player stats: {currentBoard.Player1.Name}\'s wins: {currentBoard.Player1.Wins} | {currentBoard.Player2.Name}\'s wins: {currentBoard.Player2.Wins}");  Console.WriteLine("Type \"rematch\" for a rematch or \"new\" to start new game with new players");  processGameMode(Console.ReadLine());  }  private static void endGameDraw()  {  currentBoard.TotalGames++;  Console.WriteLine($"Board is full, none of the player have won.");  Console.WriteLine("Type \"rematch\" for a rematch or \"new\" to start new game with new players");  processGameMode(Console.ReadLine());  }  private static void Rematch()  {  Console.WriteLine("Starting new round");  currentBoard.Rematch();  drawCurrentBoard();  processGame();  }  }  public class Board  {  private Player player1;  private Player player2;  public int TurnCount { get; set; }  public int TotalGames { get; set; }  public int Size { get; private set; } = 3;  public string[] CurrentState { get; set; }  public Player Player1  {  get { return player1; }  private set { player1 = value; }  }  public Player Player2  {  get { return player2; }  private set { player2 = value; }  }  public Board()  {  this.TurnCount = 0;  this.TotalGames = 0;  this.CurrentState = new string[Size \* Size];  this.player1 = new Player();  this.player2 = new Player();  this.player1.Char = "x";  this.player2.Char = "o";  }  private void fillBoard()  {  for (int i = 0; i < 9; i++) { CurrentState[i] = " "; }  }  public void Rematch()  {  this.TurnCount = 0;  this.player1.Turns = 0;  this.player2.Turns = 0;  this.player2.movesChain.Clear();  this.player2.movesChain.Clear();  fillBoard();  }  public void NewGame()  {  this.TurnCount = 0;  this.TotalGames = 0;  fillBoard();  this.player1.ResetPlayerData();  this.player1.ResetPlayerData();  Console.WriteLine("Enter Player1 name (1 char. or more): ");  player1.setName(Console.ReadLine());  Console.WriteLine($"Player 1 name is {player1.Name}");  Console.WriteLine("Enter Player2 name (1 char. or more): ");  player2.setName(Console.ReadLine());  Console.WriteLine($"Player 2 name is {player2.Name}");  Console.WriteLine($"New board prepared. Total games in session so far: {this.TotalGames}");  }  }  public class Player  {  public string Name { get; set; }  public int Turns { get; set; }  public int Wins { get; set; }  public string Char { get; set; }  public List<int> movesChain { get; set; }  public Player()  {  this.Name = "";  this.Char = "";  this.Wins = 0;  this.Turns = 0;  this.movesChain = new List<int>();  }  public void setName(string text)  {  while (text.Length < 1)  {  Console.WriteLine("Invalid Player name, try again: ");  text = Console.ReadLine();  }  this.Name = text;  }  public void ResetPlayerData()  {  this.Name = "";  this.Turns = 0;  this.Wins = 0;  this.movesChain.Clear();  }  }  } |

