|  |
| --- |
| namespace Week5GradedExercise |
|  | { |
|  | class Program |
|  | { |
|  | private static Table table = new Table(); // made the table a field so all the methods could get to it. |
|  | static void Main(string[] args) |
|  | { |
|  | try |
|  | { |
|  | Console.WriteLine("Welcome to the roulette table!"); |
|  | Roulette(); // Re-directed main method to allow catching all possible exceptions. |
|  | } |
|  | catch (Exception e) |
|  | { |
|  | Console.WriteLine(e); |
|  | } |
|  | } |
|  |  |
|  | static void Roulette() |
|  | { |
|  | Random random = new Random(); //random number generator |
|  | while (true) //open the infinite loop for user input |
|  | { |
|  | try //try-catch to keep random characters from messing up the program |
|  | { |
|  | Console.Write("Enter a number to check, N to stop, or S to spin: "); //prompt user |
|  | string more = Console.ReadLine(); //user input |
|  | if (more.ToLower() == "n") break; //if they want to quit |
|  | else if (more.ToLower() == "s") GetWins(random.Next(38)); //if they want to spin |
|  | else if (more == "00") GetWins(37); //if they enter 00 (which is 37 for our purposes) |
|  | else //entered anything other than n, s, or 00 |
|  | { |
|  | int value = int.Parse(more); //change the input to an int |
|  | if (value >= 0 && value <= 36) GetWins(value); //if int is in range |
|  | else Console.WriteLine("Enter a value between 0 and 36 or 00."); //if int is out of range |
|  | } |
|  | } |
|  | catch (Exception) //catches non-int other than n, s, and 00 |
|  | { |
|  | Console.WriteLine("Enter a value between 0 and 36 or 00."); |
|  | } |
|  |  |
|  | } |
|  | } |
|  |  |
|  | static void GetWins(int i) //returns the winning bets for the number provided (37 for 00) |
|  | { |
|  | //The first line checks if the number is 37 and returns the appropriate response for 00. |
|  | if (i == 37) Console.Write($"The following bets win on 00:\nThe Straight Up win is 00.\n"); |
|  | //The next two lines work for the rest of the numbers |
|  | else Console.Write($"The following bets win on {i}:\nThe Straight Up win is {i}.\n"); |
|  | Console.Write($"The Split wins are: {Split(i)}.\n"); |
|  |  |
|  | if (i > 0 && i < 37) //only non-zero numbers have these winning bets |
|  | { |
|  | Console.Write($"The Street win is: ({Street(i)}).\n" + |
|  | $"The Corner wins are: {Corner(i)}.\n" + |
|  | $"The Line wins are: {Line(i)}.\n" + |
|  | $"The Column win is: {Column(i)}.\n" + |
|  | $"The Dozen win is: {Dozen(i)}.\n" + |
|  | $"The Low/High number win is: {Low\_High(i)}.\n" + |
|  | $"The Odd/Even win is: {Odd\_Even(i)}.\n" + |
|  | $"The Color win is: {table.GetColor(i)}.\n"); |
|  | } |
|  | if (i < 4 || i == 37) Console.Write($"The Basket bet wins (0, 00, 1, 2, 3).\n");//checks if the basket bet wins |
|  | Console.WriteLine();//adds an extra line to seperate multiple spins |
|  | } |
|  |  |
|  | static string Split(int i)//returns numbers adjacent to the input. |
|  | { |
|  | if (table.Column(i)==0)//if the number is in the first column |
|  | { |
|  | if (table.Row(i) == 0)//if the number is in the first row |
|  | { |
|  | return $"({table.GetNum(table.Row(i), table.Column(i))}," + //the number |
|  | $" {table.GetNum(table.Row(i), (table.Column(i) + 1))})," + //to the right |
|  | $" ({table.GetNum(table.Row(i), table.Column(i))}," + //the number |
|  | $" {table.GetNum((table.Row(i) + 1), table.Column(i))})"; //below |
|  | } |
|  | else if (table.Row(i) == 11)//if the number is in the last row |
|  | { |
|  | return $"({table.GetNum((table.Row(i) - 1), table.Column(i))}," + //above |
|  | $" {table.GetNum(table.Row(i), table.Column(i))})," + //the number |
|  | $" ({table.GetNum(table.Row(i), table.Column(i))}," + //the number |
|  | $" {table.GetNum(table.Row(i), (table.Column(i) + 1))})"; //to the right |
|  | } |
|  | else |
|  | { |
|  | return $"({table.GetNum((table.Row(i) - 1), table.Column(i))}," + //above |
|  | $" {table.GetNum(table.Row(i), table.Column(i))})," + //the number |
|  | $" ({table.GetNum(table.Row(i), table.Column(i))}," + //the number |
|  | $" {table.GetNum(table.Row(i), (table.Column(i) + 1))})," +//to the right |
|  | $" ({table.GetNum(table.Row(i), table.Column(i))}," + //the number |
|  | $" {table.GetNum((table.Row(i) + 1), table.Column(i))})"; //below |
|  | } |
|  | } |
|  | else if (table.Column(i) == 1) //if the number is in the second column |
|  | { |
|  | if (table.Row(i) == 0) //the notes below here were redundant and thus omitted |
|  | { |
|  | return $"({table.GetNum(table.Row(i), (table.Column(i) - 1))}," + |
|  | $" {table.GetNum(table.Row(i), table.Column(i))})," + |
|  | $" ({table.GetNum(table.Row(i), table.Column(i))}," + |
|  | $" {table.GetNum(table.Row(i), (table.Column(i) + 1))})," + |
|  | $" ({table.GetNum(table.Row(i), table.Column(i))}," + |
|  | $" {table.GetNum((table.Row(i) + 1), table.Column(i))})"; |
|  | } |
|  | else if (table.Row(i) == 11) |
|  | { |
|  | return $"({table.GetNum((table.Row(i) - 1), table.Column(i))}," + |
|  | $" {table.GetNum(table.Row(i), table.Column(i))})," + |
|  | $" ({table.GetNum(table.Row(i), (table.Column(i) - 1))}," + |
|  | $" {table.GetNum(table.Row(i), table.Column(i))})," + |
|  | $" ({table.GetNum(table.Row(i), table.Column(i))}," + |
|  | $" {table.GetNum(table.Row(i), (table.Column(i) + 1))})"; |
|  | } |
|  | else |
|  | { |
|  | return $"({table.GetNum((table.Row(i) - 1), table.Column(i))}," + |
|  | $" {table.GetNum(table.Row(i), table.Column(i))})," + |
|  | $" ({table.GetNum(table.Row(i), (table.Column(i) - 1))}," + |
|  | $" {table.GetNum(table.Row(i), table.Column(i))})," + |
|  | $" ({table.GetNum(table.Row(i), table.Column(i))}," + |
|  | $" {table.GetNum(table.Row(i), (table.Column(i) + 1))})," + |
|  | $" ({table.GetNum(table.Row(i), table.Column(i))}," + |
|  | $" {table.GetNum((table.Row(i) + 1), table.Column(i))})"; |
|  | } |
|  | } |
|  | else if (table.Column(i) == 2) |
|  | { |
|  | if (table.Row(i) == 0) |
|  | { |
|  | return $"({table.GetNum(table.Row(i), (table.Column(i) - 1))}," + |
|  | $" {table.GetNum(table.Row(i), table.Column(i))})," + |
|  | $" ({table.GetNum(table.Row(i), table.Column(i))}," + |
|  | $" {table.GetNum((table.Row(i) + 1), table.Column(i))})"; |
|  | } |
|  | else if (table.Row(i) == 11) |
|  | { |
|  | return $"({table.GetNum((table.Row(i) - 1), table.Column(i))}," + |
|  | $" {table.GetNum(table.Row(i), table.Column(i))})," + |
|  | $" ({table.GetNum(table.Row(i), (table.Column(i) - 1))}," + |
|  | $" {table.GetNum(table.Row(i), table.Column(i))})"; |
|  | } |
|  | else |
|  | { |
|  | return $"({table.GetNum((table.Row(i) - 1), table.Column(i))}," + |
|  | $" {table.GetNum(table.Row(i), table.Column(i))})," + |
|  | $" ({table.GetNum(table.Row(i), (table.Column(i) - 1))}," + |
|  | $" {table.GetNum(table.Row(i), table.Column(i))})," + |
|  | $" ({table.GetNum(table.Row(i), table.Column(i))}," + |
|  | $" {table.GetNum((table.Row(i) + 1), table.Column(i))})"; |
|  | } |
|  | } |
|  | else //not in any of the columns |
|  | { |
|  | return "(0, 00)"; |
|  | } |
|  | } |
|  |  |
|  | static string Street(int i) //returns the numbers in the same row as the input |
|  | { |
|  | return $"{table.GetNum(table.Row(i), 0)}," + |
|  | $" {table.GetNum(table.Row(i), 1)}," + |
|  | $" {table.GetNum(table.Row(i), 2)}"; |
|  | } |
|  |  |
|  | static string Corner(int i) //returns each four number set that shares a corner with the input |
|  | { |
|  | if (table.Column(i) == 0) //same concept as Split with checking for column and row before returning results |
|  | { |
|  | if (table.Row(i) == 0) |
|  | { |
|  | return $"({table.GetNum(table.Row(i), table.Column(i))}," + |
|  | $" {table.GetNum(table.Row(i), (table.Column(i) + 1))}," + |
|  | $" {table.GetNum((table.Row(i) + 1), table.Column(i))}," + |
|  | $" {table.GetNum((table.Row(i) + 1), (table.Column(i) + 1))})"; |
|  | } |
|  | else if (table.Row(i) == 11) |
|  | { |
|  | return $"({table.GetNum((table.Row(i) - 1), table.Column(i))}," + |
|  | $" {table.GetNum((table.Row(i) - 1), (table.Column(i) + 1))}," + |
|  | $" {table.GetNum(table.Row(i), table.Column(i))}," + |
|  | $" {table.GetNum(table.Row(i), (table.Column(i) + 1))}),"; |
|  | } |
|  | else |
|  | { |
|  | return $"({table.GetNum((table.Row(i) - 1), table.Column(i))}," + |
|  | $" {table.GetNum((table.Row(i) - 1), (table.Column(i) + 1))}," + |
|  | $" {table.GetNum(table.Row(i), table.Column(i))}," + |
|  | $" {table.GetNum(table.Row(i), (table.Column(i) + 1))})," + |
|  | $" ({table.GetNum(table.Row(i), table.Column(i))}," + |
|  | $" {table.GetNum(table.Row(i), (table.Column(i) + 1))}," + |
|  | $" {table.GetNum((table.Row(i) + 1), table.Column(i))}," + |
|  | $" {table.GetNum((table.Row(i) + 1), (table.Column(i) + 1))})"; |
|  | } |
|  | } |
|  | else if (table.Column(i) == 1) |
|  | { |
|  | if (table.Row(i) == 0) |
|  | { |
|  | return $"({table.GetNum(table.Row(i), (table.Column(i) - 1))}," + |
|  | $" {table.GetNum(table.Row(i), table.Column(i))}," + |
|  | $" {table.GetNum((table.Row(i) + 1), (table.Column(i) - 1))}," + |
|  | $" {table.GetNum((table.Row(i) + 1), table.Column(i))})," + |
|  | $" ({table.GetNum(table.Row(i), table.Column(i))}," + |
|  | $" {table.GetNum(table.Row(i), (table.Column(i) + 1))}," + |
|  | $" {table.GetNum((table.Row(i) + 1), table.Column(i))}," + |
|  | $" {table.GetNum((table.Row(i) + 1), (table.Column(i) + 1))})"; |
|  | } |
|  | else if (table.Row(i) == 11) |
|  | { |
|  | return $"({table.GetNum((table.Row(i) - 1), (table.Column(i) - 1))}," + |
|  | $" {table.GetNum((table.Row(i) - 1), table.Column(i))}," + |
|  | $" {table.GetNum(table.Row(i), (table.Column(i) - 1))}," + |
|  | $" {table.GetNum(table.Row(i), table.Column(i))})," + |
|  | $" ({table.GetNum((table.Row(i) - 1), table.Column(i))}," + |
|  | $" {table.GetNum((table.Row(i) - 1), (table.Column(i) + 1))}," + |
|  | $" {table.GetNum(table.Row(i), table.Column(i))}," + |
|  | $" {table.GetNum(table.Row(i), (table.Column(i) + 1))})"; |
|  | } |
|  | else |
|  | { |
|  | return $"" + |
|  | $"({table.GetNum((table.Row(i) - 1), (table.Column(i) - 1))}," + |
|  | $" {table.GetNum((table.Row(i) - 1), table.Column(i))}," + |
|  | $" {table.GetNum(table.Row(i), (table.Column(i) - 1))}," + |
|  | $" {table.GetNum(table.Row(i), table.Column(i))})," + |
|  | $" ({table.GetNum((table.Row(i) - 1), table.Column(i))}," + |
|  | $" {table.GetNum((table.Row(i) - 1), (table.Column(i) + 1))}," + |
|  | $" {table.GetNum(table.Row(i), table.Column(i))}," + |
|  | $" {table.GetNum(table.Row(i), (table.Column(i) + 1))})," + |
|  | $" ({table.GetNum(table.Row(i), (table.Column(i) - 1))}," + |
|  | $" {table.GetNum(table.Row(i), table.Column(i))}," + |
|  | $" {table.GetNum((table.Row(i) + 1), (table.Column(i) - 1))}," + |
|  | $" {table.GetNum((table.Row(i) + 1), table.Column(i))})," + |
|  | $" ({table.GetNum(table.Row(i), table.Column(i))}," + |
|  | $" {table.GetNum(table.Row(i), (table.Column(i) + 1))}," + |
|  | $" {table.GetNum((table.Row(i) + 1), table.Column(i))}," + |
|  | $" {table.GetNum((table.Row(i) + 1), (table.Column(i) + 1))})" + |
|  | $""; |
|  | } |
|  | } |
|  | else |
|  | { |
|  | if (table.Row(i) == 0) |
|  | { |
|  | return $"({table.GetNum(table.Row(i), (table.Column(i) - 1))}," + |
|  | $" {table.GetNum(table.Row(i), table.Column(i))}," + |
|  | $" {table.GetNum((table.Row(i) + 1), (table.Column(i) - 1))}," + |
|  | $" {table.GetNum((table.Row(i) + 1), table.Column(i))})"; |
|  | } |
|  | else if (table.Row(i) == 11) |
|  | { |
|  | return $"({table.GetNum((table.Row(i) - 1), (table.Column(i) - 1))}," + |
|  | $" {table.GetNum((table.Row(i) - 1), table.Column(i))}," + |
|  | $" {table.GetNum(table.Row(i), (table.Column(i) - 1))}," + |
|  | $" {table.GetNum(table.Row(i), table.Column(i))})"; |
|  |  |
|  | } |
|  | else |
|  | { |
|  | return $"({table.GetNum((table.Row(i) - 1), (table.Column(i) - 1))}," + |
|  | $" {table.GetNum((table.Row(i) - 1), table.Column(i))}," + |
|  | $" {table.GetNum(table.Row(i), (table.Column(i) - 1))}," + |
|  | $" {table.GetNum(table.Row(i), table.Column(i))})," + |
|  | $" ({table.GetNum(table.Row(i), (table.Column(i) - 1))}," + |
|  | $" {table.GetNum(table.Row(i), table.Column(i))}," + |
|  | $" {table.GetNum((table.Row(i) + 1), (table.Column(i) - 1))}," + |
|  | $" {table.GetNum((table.Row(i) + 1), table.Column(i))})"; |
|  | } |
|  | } |
|  | } |
|  |  |
|  | static string Line(int i) |
|  | { |
|  | if (table.Row(i) == 0) |
|  | { |
|  | return $"({table.GetNum(table.Row(i), 0)}," + |
|  | $" {table.GetNum(table.Row(i), 1)}," + |
|  | $" {table.GetNum(table.Row(i), 2)}," + |
|  | $" {table.GetNum((table.Row(i) + 1), 0)}," + |
|  | $" {table.GetNum((table.Row(i) + 1), 1)}," + |
|  | $" {table.GetNum((table.Row(i) + 1), 2)})"; |
|  | } |
|  | else if (table.Row(i) == 11) |
|  | { |
|  | return $"({table.GetNum((table.Row(i) - 1), 0)}," + |
|  | $" {table.GetNum((table.Row(i) - 1), 1)}," + |
|  | $" {table.GetNum((table.Row(i) - 1), 2)}," + |
|  | $" {table.GetNum(table.Row(i), 0)}," + |
|  | $" {table.GetNum(table.Row(i), 1)}," + |
|  | $" {table.GetNum(table.Row(i), 2)})"; |
|  | } |
|  | else |
|  | { |
|  | return $"({table.GetNum((table.Row(i) - 1), 0)}," + |
|  | $" {table.GetNum((table.Row(i) - 1), 1)}," + |
|  | $" {table.GetNum((table.Row(i) - 1), 2)}," + |
|  | $" {table.GetNum(table.Row(i), 0)}," + |
|  | $" {table.GetNum(table.Row(i), 1)}," + |
|  | $" {table.GetNum(table.Row(i), 2)}), " + |
|  | $" ({table.GetNum(table.Row(i), 0)}," + |
|  | $" {table.GetNum(table.Row(i), 1)}," + |
|  | $" {table.GetNum(table.Row(i), 2)}," + |
|  | $" {table.GetNum((table.Row(i) + 1), 0)}," + |
|  | $" {table.GetNum((table.Row(i) + 1), 1)}," + |
|  | $" {table.GetNum((table.Row(i) + 1), 2)})"; |
|  | } |
|  | } |
|  |  |
|  | static string Column(int i) //returns a result based on the column of the input |
|  | { |
|  | if (table.Column(i) == 0) return "1st (1, 4, ... 31, 34)"; |
|  | else if (table.Column(i) == 1) return "2nd (2, 5, ... 32, 35)"; |
|  | else return "3rd (3, 6, ... 33, 36)"; |
|  | } |
|  |  |
|  | static string Dozen(int i) //returns which third of the numbers the input is in |
|  | { |
|  | if (i < 19 && i > 0) return "1st (1-12)"; |
|  | else if (i < 19 && i > 0) return "2nd (13-24)"; |
|  | else return "3rd (25-36)"; |
|  | } |
|  |  |
|  | static string Low\_High(int i) //returns which half of the numbers the input is in |
|  | { |
|  | if (i < 19 && i > 0) return "low (1-18)"; |
|  | else return "high (19-36)"; |
|  | } |
|  |  |
|  | static string Odd\_Even(int i) //returns odd or even based on input |
|  | { |
|  | if (i % 2 == 0) return "even"; |
|  | else return "odd"; |
|  | } |
|  | } |
|  | } |