# Documentation for main function in GithubBowling.cpp

## Overview

The main function in the GithubBowling.cpp file is the entry point of the program. This function is responsible for initializing a BowlingGame object, prompting the user for input, processing this input, updating the game state, and displaying the results. The function continues to execute until the 11th frame of the bowling game is completed.

## Return Value

The function returns an integer which represents the exit status of the program. A return value of 0 typically indicates that the program has executed successfully without any errors.

## Detailed Description

The main function begins by initializing a BowlingGame object. This object represents the state of the bowling game, including the number of pins knocked down in each roll and the total score.

Next, the function prompts the user to enter the number of pins knocked down for each roll. This input is processed and used to update the state of the BowlingGame object.

The function then displays the results of the game, including the score for each frame and the total score.

The function continues to prompt the user for input and update the game state until the 11th frame is completed. At this point, the function terminates and returns the exit status of the program.

## Usage

This function is automatically called when the program is run. The user does not need to call this function directly. Instead, the user should enter the number of pins knocked down for each roll when prompted by the function.

# Documentation for BowlingGame class in calculateScore\_Github.cpp

## Overview

The BowlingGame class in the calculateScore\_Github.cpp file represents a bowling game. It contains methods for rolling a ball, calculating the score, and checking if a frame is a strike or a spare.

## Class Members

* std::vector<int>& rolls: A reference to a vector of integers representing the number of pins knocked down in each roll.
* int currentRoll: An integer representing the current roll.

## Constructor

The BowlingGame constructor initializes the rolls vector and sets currentRoll to 0. The rolls vector is reserved for 21 elements, corresponding to the maximum number of rolls in a bowling game.

## Methods

* void roll(int pins): This method is used to record a roll in the game. It takes an integer representing the number of pins knocked down and adds it to the rolls vector.
* int score(): This method calculates and returns the total score of the game. It iterates over each frame, checks if it is a strike or a spare, and adds the appropriate score.
* bool isStrike(int frameIndex): This method checks if a frame is a strike. It takes an integer representing the frame index and returns true if the frame is a strike.
* int sumOfBallsInFrame(int frameIndex): This method calculates and returns the sum of the balls in a frame. It takes an integer representing the frame index.
* int spareBonus(int frameIndex): This method calculates and returns the bonus for a spare. It takes an integer representing the frame index.
* int strikeBonus(int frameIndex): This method calculates and returns the bonus for a strike. It takes an integer representing the frame index.
* bool isSpare(int frameIndex): This method checks if a frame is a spare. It takes an integer representing the frame index and returns true if the frame is a spare.

## Usage

To use this class, create an instance of BowlingGame, call the roll method for each roll in the game, and then call the score method to calculate the total score.

# Documentation for Unit Tests in calculateScore\_Github\_test.cpp

## Overview

The calculateScore\_Github\_test.cpp file contains unit tests for the BowlingGame class in the calculateScore\_Github.cpp file. These tests are written using the Google Test framework.

## Test Cases

**Test Case: allZeros**

This test case checks the scenario where all rolls in a game knock down zero pins. The expected score for this game is 0.

**Test Case: allOnes**

This test case checks the scenario where all rolls in a game knock down one pin. The expected score for this game is 20.

**Test Case: oneSpare**

This test case checks the scenario where the game has one spare (two rolls in a frame knock down all ten pins) followed by a roll that knocks down three pins, with all other rolls knocking down zero pins. The expected score for this game is 16.

**Test Case: oneStrike**

This test case checks the scenario where the game has one strike (a roll knocks down all ten pins) followed by two rolls that knock down three and four pins respectively, with all other rolls knocking down zero pins. The expected score for this game is 24.

**Test Case: perfectGame**

This test case checks the scenario where all rolls in a game are strikes, resulting in a perfect game. The expected score for this game is 300.

## Main Function

The main function initializes the Google Test framework and runs all test cases. It takes command line arguments argc and argv, which are passed to the testing::InitGoogleTest function. The function returns the result of RUN\_ALL\_TESTS(), which runs all test cases and returns an exit status.