**Program content:**

The program is a game simulator based on prisoner dilemma.

Prisoner dilemma: Two criminals are arrested and about to be prosecuted. However, the prosecutors lack evidence to put both of them in jail for a long period of time. Therefore he gives the two prisoners a choice to either stay silence (cooperate with the other prisoner) or sell out their mate (defect). There are 3 results:

1. If both of them keep silence, they only need to serve a light sentence.
2. If one defects while the other one keeps silence, the one who defects will go free, the one who keep silence will serve a really long jail time.
3. If both of them choose to defect, both will serve a relatively long jail time.

It is in the best interest of both prisoners to keep silence, but the prosecutor believes that they are most likely to sell out their mate to gain personal benefit. To turn this into a game, each player is awarded points based on how much benefit they receive when they decide to either cooperate or defect.

As a player, the user will be the 1st prisoner one and plays against the computer as the 2nd prisoner. The user has 6 tries. Each try, the user will either choose to cooperate or defect. The computer will choose to cooperate or defect as well. The responses will be compared and points will be given to each side following the bellow matrix:

|  |  |  |
| --- | --- | --- |
|  | Player 2 Cooperates | Player 2 Defects |
| Player 1 Cooperates | Player 1 score = 3  Player 2 score = 3 | Player 1 score = 0  Player 2 score = 5 |
| Player 1 Defects | Player 1 score = 5  Player 2 score = 0 | Player 1 score = 1  Player 2 score = 1 |

At the end of the 6th try, all the scores will be added together for both prisoners compared. Whoever has a higher score wins. Of course, you will always win if you simply defect every round, so much of the intrigue of the game comes from getting the highest score possible.

**Debugging experience:**

Delegate type cannot be declared as static.

Using an infinite while loop, it needs a condition to break out of the loop. The condition to break out of the loop need to be logical otherwise the loop will never break.

If a delegate object is passed into a method parameter and return any type (such as int, string, etc) it should be assign to a variable of the desired type before the result is being used for anything else.

There are conditions needed to force the user only input the allowed character, otherwise the program will be confused about the choice.

All static variables have to be declare outside of the main method.

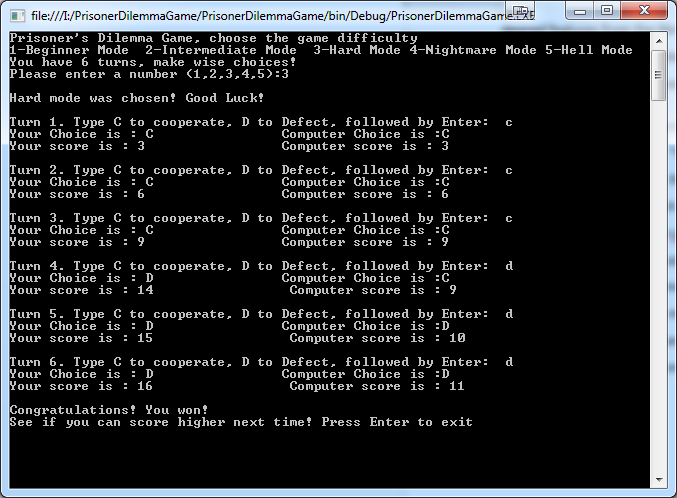
Using system and namespace are needed for the C# code to run.

**Code Implementation and Explanation:**

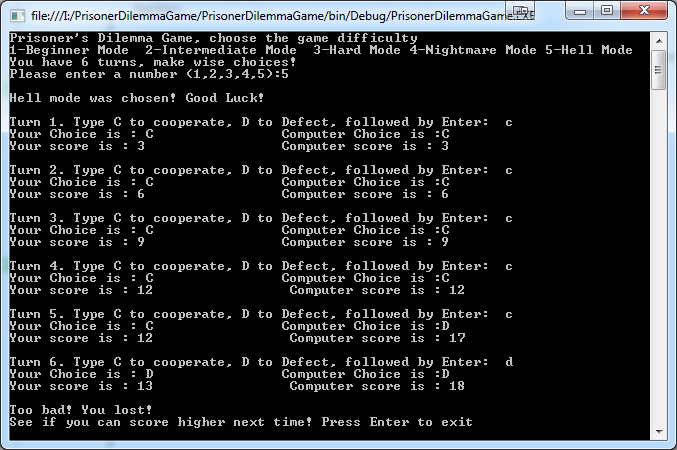
In C# many features are not imported by default so the programmer need to use headers to import the desired features from the library. The player will be asked to choose 1 of the 5 difficulty levels to play against the computer. The player choice is converted to an integer variable and used to choose which strategy the computer will use to play against the player.

Nested if statements are used to assign the strategy method to the delegate object. These strategy will return either “D”(defect) or “C”(cooperate) as the computer choice each time the game loop iterates. After the strategy is picked, the code will enter a loop where the game takes place. It will ask the players to choose “D” or “C” for each turn. The loop will show both side’s choices and show their total scores. Inside the loop, the method compareScores is used to compare the two side answer and add appropriate scores to each side integer arrays (the scores are added up each turn). The method has two parameter, one string and one delegate. Depend on the strategy has been chosen above by the computer, it will invoke the method and return appropriate result for each turn. The method return is then assigned to a variable and is compared with the player’s choice to decide the score. After 6 turns, an if statement is used to compare the score of two player and decide who is the winner.

**Screenshots:**

Game played on hard difficulty:

Game played on hell difficulty:



Game played with errors on input:

