Extras

* Clear() Method
* CurrentScore() Method to always have a scoreboard displayed at top of console
* Conversion() Method that converts gold to elixir
* Error handling for Main Methods and Extras
* Upgrade() Method for upgrading armies
* Coinflip() Method for an additional way to gain elixir
* Bonus round on wizard training that awards gold

Main

|  |  |  |
| --- | --- | --- |
| Input | Processing | Output |
| userInput | string[] deck = CreateDeck();  string currentArmy = "";  int currentElixir = 300;  int currentGold = 500;  bool upgrades = false;  CurrentScore(currentGold, currentElixir);  string userInput = GetMenuChoice(deck, ref currentArmy, ref currentElixir, ref currentGold, ref upgrades);  while(userInput!= "0"){  if(currentGold == 1500){  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("YOU REACHED 1500 GOLD!!! YOU WIN!!!");  userInput = "0";  }  else if(currentElixir == 0){  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("You ran out of elixir!!! You lose :( Try again next time!");  userInput = "0";  }  Route(userInput, deck, ref currentArmy, ref currentElixir, ref currentGold, ref upgrades);  userInput = GetMenuChoice(deck, ref currentArmy, ref currentElixir, ref currentGold, ref upgrades);  } |  |

GetMenuChoice()

|  |  |  |
| --- | --- | --- |
| Input | Processing | Output |
| deck  ref currentArmy  ref currentElixir  ref currentGold  ref upgrades | DisplayMenu(deck, ref currentArmy, ref currentElixir, ref currentGold);  return Console.ReadLine(); | Console.ReadLine() |

DisplayMenu()

|  |  |  |
| --- | --- | --- |
| Input | Processing | Output |
| deck  ref currentArmy  ref currentElixir  ref currentGold | Clear();  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("1. Wizard Training\n2. Raiding\n3. Inventory/Scoreboard\n4. Conversion\n5. Upgrades\n6. Coinflip\n0. Exit"); |  |

Route()

|  |  |  |
| --- | --- | --- |
| Input | Processing | Output |
| userInput  deck  ref currentArmy  ref currentElixir  ref currentGold  ref upgrades | switch(userInput){  case "1":  WizardTraining(deck, ref currentElixir, currentGold);  break;  case "2":  RaidSetup(ref currentArmy, ref currentElixir, ref currentGold, ref upgrades);  break;  case "3":  Scoreboard(currentGold, currentElixir);  break;  case "4":  Conversion(ref currentGold, ref currentElixir);  break;  case "5":  Upgrades(ref upgrades, ref currentGold, ref currentElixir);  break;  case "6":  CoinFlip(ref currentGold, ref currentElixir);  break;  default:  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("Invalid input!");  break;  } |  |

CreateDeck()

|  |  |  |
| --- | --- | --- |
| Input | Processing | Output |
|  | string[] suits = {"Hearts", "Diamonds", "Clubs", "Spades"};  string[] numbers = {"Ace", "2", "3", "4", "5", "6", "7", "8", "9", "10", "Jack", "Queen", "King"};  string[] deck = new string[52];  int count = 0;  for(int i = 0; i < suits.Length; i++){  for(int j = 0; j < numbers.Length; j++){  deck[count] = numbers[j] + " of " + suits[i];  count++;  }  }  return deck; | deck |

Shuffle()

|  |  |  |
| --- | --- | --- |
| Input | Processing | Output |
| deck | for(int i = 0; i < deck.Length; i++)  {  Swap(deck, GetRandomNum(), GetRandomNum());  } |  |

Swap()

|  |  |  |
| --- | --- | --- |
| Input | Processing | Output |
| myArray  int x  int y | string temp = myArray[x];  myArray[x] = myArray[y];  myArray[y] = temp; |  |

GetRandomNum()

|  |  |  |
| --- | --- | --- |
| Input | Processing | Output |
|  | Random rnd = new Random();  return rnd.Next(0, 52); | Random number from parameters given |

GetRandomRaidNum()

|  |  |  |
| --- | --- | --- |
| Input | Processing | Output |
|  | Random rnd = new Random();  return rnd.Next(0,9); | Random number from parameters given |

GetRandomCoin()

|  |  |  |
| --- | --- | --- |
| Input | Processing | Output |
|  | Random rnd = new Random();  return rnd.Next(1,3); | Random number from parameters given |

WizardTraining()

|  |  |  |
| --- | --- | --- |
| Input | Processing | Output |
| deck  ref currentElixir  ref currentGold | int currentRound = 1;  int finalRound = 0;  string[] guesses = new string[10];  int wager = 0;  Clear();  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("How much elixir do you want to wager?");  wager = int.Parse(Console.ReadLine());  while(wager > currentElixir){  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("Your wager is greater than your current elixir! Please enter a lower wager!");  wager = int.Parse(Console.ReadLine());  }  while(currentElixir - wager < 1){  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("That wager would leave your current elixir below 1! Please enter a lower wager!");  wager = int.Parse(Console.ReadLine());  }  while(currentRound != 0){  finalRound = Guessing(ref currentRound, guesses, deck, currentGold, currentElixir);  }  if(finalRound == 0){  currentElixir -= wager;  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine($"You didn't make it through a single round!!! You lost {wager} elixir!!!");  }  else if(finalRound == 1){  //no change in elixir  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine($"You made it through one round, you didn't gain or lose any elixir.");  }  else if(finalRound == 2){  currentElixir += wager;  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine($"Nice job, you made it through two rounds! You gained {wager} elixir!");  }  else if(finalRound == 3){  currentElixir += wager \* 2;  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine($"Fantastic Job!!! You made it through all three rounds!!! You gained {wager \* 2} elixir!!!");  System.Console.WriteLine($"Would you like to play the bonus round?\nYou don't have to wager anything but you can win 100 gold!!!\n1. To play the bonus round\n2. To return to main menu");  if(int.Parse(Console.ReadLine()) == 1){  BonusWizardTraining(guesses, deck, ref currentGold, ref currentElixir);  }  }  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("Would you like to continue your training or exit to the main menu?\n1. To continue\n2. For main menu");  if(int.Parse(Console.ReadLine()) == 1){  WizardTraining(deck, ref currentElixir, currentGold);  } |  |

Guesssing()

|  |  |  |
| --- | --- | --- |
| Input | Processing | Output |
| ref currentRound  guess  deck  currentGold  currentElixir | //round 1  if(currentRound == 1){  Shuffle(deck);  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine($"Cards to memorize: {deck[0]}, {deck[1]}");  Clear();  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("Enter guesses!");  guess[0] = Console.ReadLine();  guess[1] = Console.ReadLine();  //check for repeat guesses  if(guess[1] == guess[0]){  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("That guess was a repeat! Please choose another guess!");  guess[1] = Console.ReadLine();  }  //pass/fail logic  if((guess[0] == deck[0] && guess[1] == deck[1]) || (guess[0] == deck[1] && guess[1] == deck[0])){  currentRound++;  return 1;  }  else{  currentRound = 0;  return 0;  }  }  //round 2  else if(currentRound == 2){  Shuffle(deck);  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine($"Cards to memorize: {deck[0]}, {deck[1]}, {deck[2]}");  Thread.Sleep(1000);  Clear();  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("Enter guesses!");  //start guesses  guess[0] = Console.ReadLine();  guess[1] = Console.ReadLine();  while(guess[1] == guess[0]){  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("That guess was a repeat! Please choose another guess!");  guess[1] = Console.ReadLine();  }  guess[2] = Console.ReadLine();  while(guess[2] == guess[0] || guess[2] == guess[1]){  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("That guess was a repeat! Please choose another guess!");  guess[2] = Console.ReadLine();  }  //pass/fail logic  if((guess[0] == deck[0] || guess[0] == deck[1] || guess[0] == deck[2]) && (guess[1] == deck[0] || guess[1] == deck[1] || guess[1] == deck[2]) && (guess[2] == deck[0] || guess[2] == deck[1] || guess[2] == deck[2])){  currentRound++;  return 2;  }  else{  currentRound = 0;  return 1;  }  }  //round 3  else if(currentRound == 3){  Shuffle(deck);  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine($"Cards to memorize: {deck[0]}, {deck[1]}, {deck[2]}, {deck[3]}, {deck[4]}");  Clear();  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("Enter guesses!");  //start guesses  guess[0] = Console.ReadLine();  guess[1] = Console.ReadLine();  while(guess[1] == guess[0]){  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("That guess was a repeat! Please choose another guess!");  guess[1] = Console.ReadLine();  }  //guess3  guess[2] = Console.ReadLine();  while(guess[2] == guess[0] || guess[2] == guess[1]){  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("That guess was a repeat! Please choose another guess!");  guess[2] = Console.ReadLine();  }  //guess4  guess[3] = Console.ReadLine();  while(guess[3] == guess[0] || guess[3] == guess[1] || guess[3] == guess[2]){  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("That guess was a repeat! Please choose another guess!");  guess[3] = Console.ReadLine();  }  //guess5  guess[4] = Console.ReadLine();  while(guess[4] == guess[0] || guess[4] == guess[1] || guess[4] == guess[2] || guess[4] == guess[3]){  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("That guess was a repeat! Please choose another guess!");  guess[4] = Console.ReadLine();  }  //pass/fail logic  if((guess[0] == deck[0] || guess[0] == deck[1] || guess[0] == deck[2] || guess[0] == deck[3] || guess[0] == deck[4])  && (guess[1] == deck[0] || guess[1] == deck[1] || guess[1] == deck[2] || guess[1] == deck[3] || guess[1] == deck[4])  && (guess[2] == deck[0] || guess[2] == deck[1] || guess[2] == deck[2] || guess[2] == deck[3] || guess[2] == deck[4])  && (guess[3] == deck[0] || guess[3] == deck[1] || guess[3] == deck[2] || guess[3] == deck[3] || guess[3] == deck[4])  && (guess[4] == deck[0] || guess[4] == deck[1] || guess[4] == deck[2] || guess[4] == deck[3] || guess[4] == deck[4]))  {  currentRound = 0;  return 3;  }  else{  currentRound = 0;  return 2;  }  }  //not needed but gets rid of CS0161  return 0; | value for  finalRound  in Wizard  Training |

RaidSetup()

|  |  |  |
| --- | --- | --- |
| Input | Processing | Output |
| ref currentArmy  ref currentElixir  ref currentGold  ref upgrades | //Gold City - 20% = Silly Goblins = 100 elixir cost  //Walled Garden - 30% = Stampeding Hog Riders = 150 elixir cost  //Mage Palace - 50% = Witty Wizards = 250 elixir cost  //if army takes elixir to 0 user should be notified and not allowed to purchase army  int choice = 0;  Clear();  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("Please choose your army:\nSilly Goblins costs 100 elixir\nStampeding Hog Riders costs 150 elixir\nWitty Wizards costs 250 elixir");  currentArmy = Console.ReadLine();  if(currentArmy == "Silly Goblins"){  if((currentElixir - 100) < 1){  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("Buying that army would leave your current elixir below 1! Cannot complete purchase!");  System.Console.WriteLine("Returning to main menu!");  currentArmy = "";  }  else{  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("Do you want to purchase this army?\n1 for yes\n2 for no");  choice = int.Parse(Console.ReadLine());  if(choice == 1){  currentElixir -= 100;  Raid(ref currentArmy, ref currentGold, currentElixir, ref upgrades);  }  else if(choice == 2){  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("Would you like to choose another army or return to main menu?\n1. for choose new army\n2. for main menu");;  choice = int.Parse(Console.ReadLine());  if(choice == 1){  RaidSetup(ref currentArmy, ref currentElixir, ref currentGold, ref upgrades);  }  else if(choice == 2){  currentArmy = "";  }  }  }  }  else if(currentArmy == "Stampeding Hog Riders"){  if((currentElixir - 150) < 1){  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("Buying that army would leave your current elixir below 1! Cannot complete purchase!");  System.Console.WriteLine("Returning to main menu!");  currentArmy = "";  }  else{  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("Do you want to purchase this army?\n1 for yes\n2 for no");  choice = int.Parse(Console.ReadLine());  if(choice == 1){  currentElixir -= 150;  Raid(ref currentArmy, ref currentGold, currentElixir, ref upgrades);  }  else if(choice == 2){  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("Would you like to choose another army or return to main menu?\n1. for choose new army\n2. for main menu");  choice = int.Parse(Console.ReadLine());  if(choice == 1){  RaidSetup(ref currentArmy, ref currentElixir, ref currentGold, ref upgrades);  }  else if(choice == 2){  currentArmy = "";  }  }  }  }  else if(currentArmy == "Witty Wizards"){  if((currentElixir - 250) < 1){  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("Buying that army would leave your current elixir below 1! Cannot complete purchase!");  System.Console.WriteLine("Returning to main menu!");  currentArmy = "";  }  else{  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("Do you want to purchase this army?\n1 for yes\n2 for no");  choice = int.Parse(Console.ReadLine());  if(choice == 1){  currentElixir -= 250;  Raid(ref currentArmy, ref currentGold, currentElixir, ref upgrades);  }  else if(choice == 2){  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("Would you like to choose another army or return to main menu?\n1. for choose new army\n2. for main menu");  choice = int.Parse(Console.ReadLine());  if(choice == 1){  RaidSetup(ref currentArmy, ref currentElixir, ref currentGold, ref upgrades);  }  else if(choice == 2){  currentArmy = "";  }  }  }  } |  |

Raid()

|  |  |  |
| --- | --- | --- |
| Input | Processing | Output |
| ref currentArmy  ref currentGold  currentElixir  ref upgrades | //Gold City - 20% = Silly Goblins = 100 elixir cost  //Walled Garden - 30% = Stampeding Hog Riders = 150 elixir cost  //Mage Palace - 50% = Witty Wizards = 250 elixir cost  //if army takes elixir to 0 user should be notified and bot allowed to purchase army  string[] raidLocations = {"Gold City", "Gold City", "Walled Garden", "Walled Garden", "Walled Garden", "Mage Palace", "Mage Palace", "Mage Palace", "Mage Palace", "Mage Palace",};  string currRaid = raidLocations[GetRandomRaidNum()];  Clear();  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine($"Current upgrade status is: {upgrades}");  System.Console.WriteLine($"The current raid is at {currRaid}!");  if(upgrades == true){  if((currentArmy == "Silly Goblins" && currRaid == "Gold City") || (currentArmy == "Silly Goblins" && currRaid == "Mage Palace")){  currentGold += 300;  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine($"The raid was successful!!! Your current gold is now {currentGold}!!!");  }  else if((currentArmy == "Stampeding Hog Riders" && currRaid == "Walled Garden") || (currentArmy == "Stampeding Hog Riders" && currRaid == "Gold City")){  currentGold += 300;  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine($"The raid was successful!!! Your current gold is now {currentGold}!!!");  }  else if((currentArmy == "Witty Wizards" && currRaid == "Mage Palace") || (currentArmy == "Witty Wizards" && currRaid == "Walled Garden")){  currentGold += 300;  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine($"The raid was successful!!! Your current gold is now {currentGold}!!!");  }  else{  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine($"The raid was unsuccessful! Your current gold is {currentGold}!");  }  }  else{  if(currentArmy == "Silly Goblins" && currRaid == "Gold City"){  currentGold += 300;  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine($"The raid was successful!!! Your current gold is now {currentGold}!!!");  }  else if(currentArmy == "Stampeding Hog Riders" && currRaid == "Walled Garden"){  currentGold += 300;  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine($"The raid was successful!!! Your current gold is now {currentGold}!!!");  }  else if(currentArmy == "Witty Wizards" && currRaid == "Mage Palace"){  currentGold += 300;  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine($"The raid was successful!!! Your current gold is now {currentGold}!!!");  }  else{  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine($"The raid was unsuccessful! Your current gold is {currentGold}!");  }  } |  |

Scoreboard()

|  |  |  |
| --- | --- | --- |
| Input | Processing | Output |
| currentGold  currentElixir | Clear();  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine($"Current Gold is {currentGold}!\nCurrnet Elixir is {currentElixir}!"); |  |

EXTRAS

Clear()

|  |  |  |
| --- | --- | --- |
| Input | Processing | Output |
|  | System.Console.WriteLine($"Press any key to continue");  Console.ReadKey();  Console.Clear(); |  |

CurrentScore()

|  |  |  |
| --- | --- | --- |
| Input | Processing | Output |
| currentGold  currentElixir | Console.SetCursorPosition(50, Console.CursorTop);  System.Console.WriteLine($"Gold: {currentGold}\tElixir: {currentElixir}"); |  |

Conversion()

|  |  |  |
| --- | --- | --- |
| Input | Processing | Output |
| ref currentGold  ref currentElixir | int goldToConvert = 0;  Clear();  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine($"You currently have {currentGold} gold and {currentElixir} elixir.\nThe conversion rate is 5 elixir per 1 gold.\nHow much gold do you want to convert?");  goldToConvert = int.Parse(Console.ReadLine());  while(currentGold - goldToConvert < 0){  System.Console.WriteLine($"That conversion would leave you with negative gold!!! Please convert less gold!!!");  goldToConvert = int.Parse(Console.ReadLine());  }  currentGold -= goldToConvert;  currentElixir += goldToConvert \* 5;  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine($"After converting your new totals are:\nGold: {currentGold} and Elixir: {currentElixir}"); |  |

Upgrades()

|  |  |  |
| --- | --- | --- |
| Input | Processing | Output |
| ref upgrades  ref currentGold  ref currentElixir | Clear();  if(upgrades == false){  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("You can pay 500 gold to upgrade your armies!\nUpgrading benefits:\nSilly Goblins raid Gold City and Mage Palace\nStampeding Hog Riders raid Walled Garden and Gold City\nWitty Wizards raid Mage Palace and Walled Garden");  System.Console.WriteLine("Do you want to upgrade?\n1. For yes\n2. For no");  int choice = int.Parse(Console.ReadLine());  if(choice == 1){  if(currentGold - 500 >= 0){  upgrades = true;  currentGold -= 500;  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("Upgrades completed successfully!\nReturning to main menu");  }  else{  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("Upgrading would leave your current gold below 0!!!\nUpgrade cancelled!!!\nReturning to main menu");  }  }  else if(choice == 2){  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("Upgrades cancelled :(\nReturning to main menu");  }  }  else{  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("Upgrades could not be completed!!!\nReturning to main menu");  } |  |

BonusWizardTraining()

|  |  |  |
| --- | --- | --- |
| Input | Processing | Output |
| guess  deck  ref currentGold  ref currentElixir | Clear();  Shuffle(deck);  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine($"Cards to memorize: {deck[0]}, {deck[1]}, {deck[2]}, {deck[3]}, {deck[4]}, {deck[5]}, {deck[6]}, {deck[7]}");  Clear();  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("Enter guesses!");  //start guesses  guess[0] = Console.ReadLine();  guess[1] = Console.ReadLine();  while(guess[1] == guess[0]){  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("That guess was a repeat! Please choose another guess!");  guess[1] = Console.ReadLine();  }  //guess3  guess[2] = Console.ReadLine();  while(guess[2] == guess[0] || guess[2] == guess[1]){  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("That guess was a repeat! Please choose another guess!");  guess[2] = Console.ReadLine();  }  //guess4  guess[3] = Console.ReadLine();  while(guess[3] == guess[0] || guess[3] == guess[1] || guess[3] == guess[2]){  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("That guess was a repeat! Please choose another guess!");  guess[3] = Console.ReadLine();  }  //guess5  guess[4] = Console.ReadLine();  while(guess[4] == guess[0] || guess[4] == guess[1] || guess[4] == guess[2] || guess[4] == guess[3]){  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("That guess was a repeat! Please choose another guess!");  guess[4] = Console.ReadLine();  }  //guess6  guess[5] = Console.ReadLine();  while(guess[5] == guess[0] || guess[5] == guess[1] || guess[5] == guess[2] || guess[5] == guess[3] || guess[5] == guess[4]){  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("That guess was a repeat! Please choose another guess!");  guess[5] = Console.ReadLine();  }  //guess7  guess[6] = Console.ReadLine();  while(guess[6] == guess[0] || guess[6] == guess[1] || guess[6] == guess[2] || guess[6] == guess[3] || guess[6] == guess[4] || guess[6] == guess[5]){  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("That guess was a repeat! Please choose another guess!");  guess[6] = Console.ReadLine();  }  //guess8  guess[7] = Console.ReadLine();  while(guess[7] == guess[0] || guess[7] == guess[1] || guess[7] == guess[2] || guess[7] == guess[3] || guess[7] == guess[4] || guess[7] == guess[5] || guess[7] == guess[6]){  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("That guess was a repeat! Please choose another guess!");  guess[7] = Console.ReadLine();  }  //pass/fail logic  if((guess[0] == deck[0] || guess[0] == deck[1] || guess[0] == deck[2] || guess[0] == deck[3] || guess[0] == deck[4] || guess[0] == deck[5] || guess[0] == deck[6] || guess[0] == deck[7])  && (guess[1] == deck[0] || guess[1] == deck[1] || guess[1] == deck[2] || guess[1] == deck[3] || guess[1] == deck[4] || guess[1] == deck[5] || guess[1] == deck[6] || guess[1] == deck[7])  && (guess[2] == deck[0] || guess[2] == deck[1] || guess[2] == deck[2] || guess[2] == deck[3] || guess[2] == deck[4] || guess[2] == deck[5] || guess[2] == deck[6] || guess[2] == deck[7])  && (guess[3] == deck[0] || guess[3] == deck[1] || guess[3] == deck[2] || guess[3] == deck[3] || guess[3] == deck[4] || guess[3] == deck[5] || guess[3] == deck[6] || guess[3] == deck[7])  && (guess[4] == deck[0] || guess[4] == deck[1] || guess[4] == deck[2] || guess[4] == deck[3] || guess[4] == deck[4] || guess[4] == deck[5] || guess[4] == deck[6] || guess[4] == deck[7])  && (guess[5] == deck[0] || guess[5] == deck[1] || guess[5] == deck[2] || guess[5] == deck[3] || guess[5] == deck[4] || guess[5] == deck[5] || guess[5] == deck[6] || guess[5] == deck[7])  && (guess[6] == deck[0] || guess[6] == deck[1] || guess[6] == deck[2] || guess[6] == deck[3] || guess[6] == deck[4] || guess[6] == deck[5] || guess[6] == deck[6] || guess[6] == deck[7])  && (guess[7] == deck[0] || guess[7] == deck[1] || guess[7] == deck[2] || guess[7] == deck[3] || guess[7] == deck[4] || guess[7] == deck[5] || guess[7] == deck[6] || guess[7] == deck[7])  ){  currentGold += 100;  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine($"YOU WON THE BONUS ROUND!!!!! You got 100 gold!! Your current gold is now {currentGold}!!!");  }  else{  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("You didn't win the bonus :(\nReturning to main menu");  } |  |

CoinFlip()

|  |  |  |
| --- | --- | --- |
| Input | Processing | Output |
| ref currentGold  ref currentElixir | int wager = 0;  Clear();  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("How much elixir do you want to wager on this coin flip? You can double it!");  wager = int.Parse(Console.ReadLine());  while(wager > currentElixir){  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("Your wager is greater than your current elixir! Please enter a lower wager!");  wager = int.Parse(Console.ReadLine());  }  while(currentElixir - wager < 1){  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("That wager would leave your current elixir below 1! Please enter a lower wager!");  wager = int.Parse(Console.ReadLine());  }  Clear();  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine($"Please enter 1 for heads or 2 for tails");  int guess = int.Parse(Console.ReadLine());  if(guess == 1 && GetRandomCoin() == 1){  currentElixir += wager;  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine($"You guessed heads and it was correct!!! Your new total of elixir is {currentElixir}");  }  else if(guess == 2 && GetRandomCoin() == 2){  currentElixir += wager;  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine($"You guessed tails and it was correct!!! Your new total of elixir is {currentElixir}");  }  else{  currentElixir -= wager;  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine($"You guessed wrong :(\nYour new elixir is {currentElixir}");  }  CurrentScore(currentGold, currentElixir);  System.Console.WriteLine("Would you like to continue your training or exit to the main menu?\n1. To continue\n2. For main menu");  if(int.Parse(Console.ReadLine()) == 1){  CoinFlip(ref currentGold, ref currentElixir);  } |  |