```cpp
/*
Daniel Avila April 15th, 2020 Section 19
Lab 10: Smart Pointers
Description: To use smart pointers to match dice and use its functions
Description: to manage the object and destroy the object
*/
#include "Dice.h"//to include the class

void checkPtrs(shared_ptr<Dice>& p1, shared_ptr<Dice>& p2);//prototype
void ptrRef(shared_ptr<Dice>& p1, shared_ptr<Dice>& p2);//prototype

int main()
{
    int diceReroll;

    cout << "===Creating Shared Pointer #1 to manage the object===" << endl;
    shared_ptr<Dice> p1(new Dice());//creating first smart pointer
    p1->displayDice();//displays the Dice's firsts values
    p1->diceMatch();//determines how many dices match

    cout << "===Now creating Shared Pointer #2 to manage the same object===" <<
      endl;
    shared_ptr<Dice> p2 = p1;//creates second smart pointer and sets it equal to
      the first
    cout << "Pick a dice to reroll" << endl;
    cin >> diceReroll;//dice option to change
    p2->reRoll(diceReroll);//dice to reroll in the class
    p2->displayDice();//displays the new value of chosen dice along with the other
       2
    p2->diceMatch();//determines if any dice match

    cout << "==Checking if pointers are being utilized==" << endl;
    checkPtrs(p1, p2);//if the pointer is pointing to any objects
    ptrRef(p1, p2);//how many times the pointers have been referenced

    cout << "===Releasing Pointer #1===" << endl;
    p1.reset();//reset the number of references the pointer has
    cout << "==Checking if pointers are being utilized==" << endl;
    checkPtrs(p1, p2);//if the pointer is pointing to any objects
    ptrRef(p1, p2);//how many times the pointers have been referenced

    cout << "===Releasing Pointer #2===" << endl;
    p2.reset();//resets the second smart pointer's pointing of an object
    cout << "==Checking if pointers are being utilized==" << endl;
    checkPtrs(p1, p2);//if the pointer is pointing to any objects
    ptrRef(p1, p2);//how many times the pointers have been referenced

    system("pause>nul");
    return 0;
}
void checkPtrs(shared_ptr<Dice>& p1, shared_ptr<Dice>& p2)//passed by reference
{//function used to check if the pointers are pointing to an object
```

```
50      if (p1)//first pointer
51          cout << "Ptr 1 currently points to an object" << endl;
52      else
53          cout << "Ptr 1 currently points to no object" << endl;
54      if (p2)//second pointer
55          cout << "Ptr 2 currently points to an object" << endl;
56      else
57          cout << "Ptr 2 currently points to no object" << endl;
58  }
59  void ptrRef(shared_ptr<Dice>& p1, shared_ptr<Dice>& p2)//passed by reference
60  {//used to check how many references each pointer has using use_count()
61      cout << endl;
62      cout << "Ptr 1's # of references in shared grouping: " << p1.use_count() <<
          endl;//number of times it is referenced
63      cout << "Ptr 2's # of references in shared grouping: " << p2.use_count() <<
          endl;
64      cout << endl;
65  }
```

```h
#ifndef DICE_H
#define DICE_H

#include <iostream>
#include <memory>//used for pointers
#include <time.h>//for random number to assign
using namespace std;

class Dice
{
private:
    int dice1;
    int dice2;
    int dice3;
public:
    Dice();
    ~Dice();
    void displayDice();
    void diceMatch();
    void reRoll(int num);
};
Dice::Dice()
{
    srand(time(0));//seed a random number
    dice1 = rand() % 6 + 1;//1-6
    dice2 = rand() % 6 + 1;//1-6
    dice3 = rand() % 6 + 1;//1-6
}
Dice::~Dice()
{//Deconstructor for when the object is not being pointed at
    cout << "No shared Pointers left to manage dice object, ";
    cout << "Deconstructor called on dice object!" << endl;
}
void Dice::displayDice()
{//Displays the values of each dice
    cout << "Dice #1 is " << dice1 << endl;
    cout << "Dice #2 is " << dice2 << endl;
    cout << "Dice #3 is " << dice3 << endl;
}
void Dice::diceMatch()
{//checks to see if any of the die match to one another
    if ((dice1 == dice2) && (dice2 == dice3) && (dice3 == dice1))
    {//for all three matching
        cout << "All Three Dice Are Equal" << endl << endl;
    }
    else if ((dice1 == dice2) || (dice1 == dice3) || (dice2 == dice3))
    {//for 2 of 3 matching
        cout << "Two Dice Are Equal" << endl << endl;
    }
    else
    {//nothing matches
        cout << "No Dice Are Equal" << endl << endl;
```

```
53      }
54
55  }
56  void Dice::reRoll(int num)
57  {
58      srand(time(0));//seed a random number
59      if (num == 1)
60      {//first dice is re-rolled
61          dice1 = rand() % 6 + 1;
62      }
63      else if (num == 2)
64      {//second dice is re-rolled
65          dice2 = rand() % 6 + 1;
66      }
67      else if (num == 3)
68      {//third die is re-rolled
69          dice3 = rand() % 6 + 1;
70      }
71  }
72
73  #endif // !DICE_H
74
```