```cpp
/*****************************************************************************
 * PROGRAMMER : Ali Eshghi
 * STUDENT ID : 1112261
 * CLASS      : CS1C
 * SECTION    : MW 5pm
 * Assign #1  : Deck of cards
 * DUE DATE   : 22 January 2020
 *****************************************************************************/

#include"Header.h"


/*****************************************************************************
 * Methods for class Deck
 *****************************************************************************/


/*************************************************************
 * Deck();
 * Constructor: initializes the count of the shuffles
 * Parameters: shuffleCount
 * Return: none
 *************************************************************/
Deck::Deck()
{
    shuffleCount = 0;
}


/*************************************************************
 * ~Deck();
 * Constructor: performs nothing
 * Parameters: none
 * Return: none
 *************************************************************/
Deck::~Deck() {}

/*************************************************************
 * void Initialize();
 * This method will initialize a new deck of cards.
 -------------------------------------------------------------
 * Parameter:
 -------------------------------------------------------------
 * Return: none
*************************************************************/
void Deck::Initialize()
{
    int index;

    //all the face values in an string array
    string faces[] = {"Ace", "2", "3", "4", "5", "6", "7", "8", "9", "10", "Jack",
"Queen", "King"};

    //put all the suit values in an array as strings
    string suits[] = {"Hearts", "Diamonds", "Clubs", "Spades"};

    for(int i = 0; i < 4; i++)
    {
        for(int j = 0; j < 13; j++)
```

```
        {
            index = (i * 13) +j;

            deck[index].suit = suits[i];
            deck[index].rank = faces[j];
        }
    }
}


/*****************************************************************
 * void Print() const;
 * This method will print deck of cards.
 ----------------------------------------------------------------
 * Parameter:
 ----------------------------------------------------------------
 * Return: none
*****************************************************************/
void Deck::print() const
{
    for(int i = 0; i < DECK_SIZE; i++)
    {
        cout << deck[i].rank << " of " << deck[i].suit << endl;
    }

    cout << endl << endl;

}

/*****************************************************************
 * void shuffle();
 * This method will shuffle the deck of cards.
 ----------------------------------------------------------------
 * Parameter:
 ----------------------------------------------------------------
 * Return: none
*****************************************************************/
void Deck::shuffle()
{
    shuffleCount++;

    Card temp[DECK_SIZE];

    for(int i = 0; i < DECK_SIZE; i += 2)
    {
        temp[i] = deck[i/2];
    }

    for(int j = 1; j < DECK_SIZE; j += 2)
    {
        temp[j] = deck[j/2 + DECK_SIZE/2];
    }

    for(int k = 0; k < DECK_SIZE; k++)
    {
        shuffled[k] = temp[k];
        deck[k] = temp[k];
    }

```

```
118 }
119
120 /*****************************************************************
121  * bool compare();
122  * This method will compare the two deck of cards
123 -----------------------------------------------------------------
124  * Parameter:
125 -----------------------------------------------------------------
126  * Return: cmpr (bool type variable)
127 *****************************************************************/
128 bool Deck::compare() const
129 {
130     bool cmpr;
131
132     Deck card;
133
134     card.Initialize();
135
136     for(int i = 0; i < DECK_SIZE; i++)
137     {
138         if((deck[i].suit == shuffled[i].suit)
139         &&  deck[i].rank == shuffled[i].rank)
140         {
141             cmpr = true;
142         }
143
144         else
145         {
146             cmpr = false;
147         }
148
149     }
150
151     return cmpr;
152 }
153
154 /*****************************************************************
155  * void returnToOriginal();
156  * This method will prints out how many shuffles needed to
157 -----------------------------------------------------------------
158  * Parameter:
159 -----------------------------------------------------------------
160  * Return: none
161 *****************************************************************/
162 void Deck::returnToOriginal() const
163 {
164     cout << shuffleCount << "  shuffles needed to return the deck to "
165          << "its original form" << endl;
166 }
167
168
169
```