```
/*
Algorithms 9/28/18
Project #2 card.h file
Flipcard Game
Rebekah Davis, Andrea Matellian, and Nathan Newbury
Group ID: DAVMATNEW
*/

#include
#include
#include
#include
#include

using namespace std;

#ifndef CARD_H
#define CARD_H
class card
{
public:
card();
card(int v, string s); //value and suit
card(const card &c);
void setValue(int);
void setSuit(string);
int getValue();
string getSuit();
friend ostream& operator<< (ostream &ostr, const card& c);
//friend bool operator== (const card& lhs, const card& rhs);
card& operator = (card &c);

  private:
    int value;
    string suit;

};

template
class node
{
```

public:
T nodeValue;
node* next;
node() : next(NULL) {}
node(const T& item, node *nextNode = NULL) :
nodeValue(item), next(nextNode) {}
};

/*
template
class dnode
{
public:
T nodeValue;
dnode *prev;
dnode *next;

```
dnode()
    {
        next = this;
        prev = this;
    }

dnode(const T& value): nodeValue(value)
    {
        next = this;
        prev = this;
    }
```

};
*/

#endif
// *** // card class implementation //** *

//empty constructor
card::card() {}

//create card given value & suit
card::card(int v, string s): value(v), suit(s)
{
setValue(v);
setSuit(s);

```
}

//copy constructor, = overloaded
card::card(const card &c)
{
card newCard = new card;
const card &c1 = c;
}

//overloaded cout to print card
ostream & operator<<(ostream & os, const card & c)
{
if(c.value == 1)
os << "card value: " << "Ace" << " ";
else if(c.value <= 10)
os << "card value: " << c.value << " ";
else if(c.value == 11)
os << "card value: " << "Jack" << " ";
else if(c.value == 12)
os << "card value: " << "Queen" << " ";
else if(c.value == 13)
os << "card value: " << "King" << " ";
os << "suit: " << c.suit << endl;
return os;
}

/*
bool operator== (const card& lhs, const card& rhs)
{
```

```
if(lhs.value == rhs.value && lhs.suit ==rhs.suit )
{
    return true;
    cout<< rhs.value<<endl;
}
else
{
    return false;
}
```

```
}
*/
```

```
void card::setValue(int v)
{
value = v;
}

void card::setSuit(string s)
{
suit = s;
}

int card::getValue()
{
return value;
}

string card::getSuit()
{
return suit;
}

//overloaded operator to create new card in copy constructor
card& card::operator = (card &c)
{
```

```
  newCard->value = c.value;
  newCard->suit = c.suit;
  return *this;
```

```
}
```