The Card class contains a mysterious item whose signature is as follows.

ostream& operator<<( ostream &stream, const Card &card )

This is an example of *operator overloading* which is the topic for next week, so don’t worry about trying to understand it. It’s purpose is so that you can print out a card using *cout* and the usual operator <<. For example, executing

Card c = Card( ACE, SPADES );

cout << c << endl;

Should produce the output line “Ace of Spades” All you need to do to get this to work is to define the static strings in the arrays “suit\_str[]” and “rank\_str[]”. Static data members cannot be defined inside the class (exception: static integers can be defined directly in the header!) but instead need to be defined at global scope in the file Card.cpp. To define them at global scope, you need to precede the array names with the scope resolution operator“Card::” Hence, you define these strings in the file Card.cpp as follows.

char \*Card::rank\_str[15] = { "", "", "two", "three", . . . "king", "ace" };

The indexes of the arrays should correspond to your enums. For example, if you defined your enum as

enum Rank {ACE = 1, TWO, THREE, FOUR, . . ., JACK, QUEEN, KING};

Then the string for ACE should be in rank\_str[1], the string for TWO should be in rank\_str[2], and so on. In this case, the static definition of the strings would be

char \*Card::rank\_str[14] = { "", "ace", "two", "three", . . . "king" };