Draw Bot

This is a **pair** exercise and must be competed in your **tutorial** or **lab** with your partner.

For this activity, you'll be creating a basic A.I. for the Final Card-Down.

We'll start out with the most basic A.I. possible: one that simply draws a card, and then ends their turn, called Draw Bot.

Remember that it's always valid at the start of the game to draw a card (whether or not you have a valid card that you could play).

Write your first A.I. for the Final Card-Down in a file called `drawBot.c`.

Base your code on the provided `drawBot.c` stub code:

Download `drawBot.c`, or copy it into your current directory on a CSE system by running

```
$ cp /web/cs1511/17s2/week12/files/drawBot.c .
```

However, Draw Bot isn't a very interesting A.I. – since it only ever draws cards, and never plays cards, it doesn't stand a chance of winning the game.

So, in this activity, you'll be trying to make Draw Bot a little bit more interesting....

We have provided you with six stubs for helper functions that will help Draw Bot play the Final Card-Down more successfully:

```
static int findMatchingCardColor (Game game, color color);
```

The `findMatchingCardColor` function finds a card in the player's hand that matches the specified color, if such a card exists.

It returns the card index, or NOT_FOUND if no matching card was found.

```
static int doCardsMatch (Card first, Card second);
```

The `doCardsMatch` function compares two cards to determine whether they match on at least one of color, suit, or value.

It returns TRUE if they match on any of the above features, and FALSE if they do not match on any of the above features.

```
static int canDrawCard (Game game);
```

The `canDrawCard` function determines whether the player can currently draw a card.

It returns TRUE if they can, and FALSE if they can't.

(If they can't draw a card, they should probably end their turn...)

```
static int shouldSayUNO (Game game);  static int shouldSayDUO (Game game);  static int shouldSayTRIO (Game game);
```

The `shouldSayUNO`, `shouldSayDUO`, and `shouldSayTRIO` functions determine whether the current player should SAY_UNO / SAY_DUO / SAY_TRIO.

There are two different situations where it could be a valid move to `SAY_UNO` (or DUO or TRIO) – for now just deal with the simple situation: "claim card". (see the diagram for more details).

Note: there are several different approaches you could take to determine whether or not the player should currently SAY_UNO / SAY_DUO / SAY_TRIO.

Once you have completed these functions, you might want to put together a better version of Draw Bot, which can do more than just drawing cards.

A few suggested strategies you could implement using the functions above:

- Is there a card in my hand that matches the top card on the discard pile?
  - If yes, then play it.
- Should I say UNO/DUO/TRIO?
  - If yes, then say it.

Do not include a main function in your file.

To run some simple automated tests:

```
$ 1511 autotest drawBot
```

To run Styl-o-matic:

```
$ 1511 stylomatic drawBot.c
Looks good!
```

You'll get advice if you need to make changes to your code.

Submit your work with the *give* command, like so:

```
$ give cs1511 wk12_drawBot
```

Or, if you are working from home, upload the relevant file(s) to the `wk12_drawBot` activity on Give Online.