COMP1511 18s2

Week-11 Tutorial Exercises

Final Card Down: Game ADT

There are three stages in the Final Card-Down assignment:

- testGame.c Testing the Game ADT,
- Game.c Implementing the Game ADT,
- player.c Using the Game ADT.

In this tutorial we will be looking at implementing the Game ADT. Please see the list of toipcs (at the end of this document) you need to cover for this week.

You should be familiar with `Game.h` now – if not, go and read it ASAP.

In the past, when we've created a struct and a pointer to a struct, we've done them in the same place, e.g.

```
typedef struct _node *Node;

typedef struct _node {
    int value;
    Node next;
}
```

In `Game.h`, you'll notice that the typedef declaring that a `Game` is a pointer to a `struct _game`, but the game struct itself isn't actually defined.

This is intentional – as it's up to you to decide how you'll implement your game struct – i.e. what fields it will need to contain. To do this, you will need to think about the game, and look at the various interface functions in `Game.h` – what will you need to store, to keep track of the game state throughout the game?

Let's think about a very *very* basic version of the game – one which has no functionality except for keeping track of turn numbers.

So, we should be able to call the `newGame` function, to create a new game; the `currentTurn` function, to get the current turn, and the `playMove` function (ignoring the actual move that's passed in).

Some example tests for this overly-simplified Game ADT implementation:

```
// Create a new game.
Game g = newGame(/* ... */);

// The turn number should start at 0.
assert (currentTurn(g) == 0);

// Make a move....
// We don't actually care what the contents of this are,
// so we'll make everything 0 for now.
playerMove move = { 0 };
move->action = END_TURN;

// Make the move
playMove(g, move);

// Because we made an END_TURN move, the turn has ended,
// and so we should now be up to turn 1 (player 1's turn)
assert (currentTurn(g) == 1);

// ... and so on.
```

What would you need in the game struct for this?

```
typedef struct _game {
    int turnNumber; // perhaps something like this?
    // ... anything else?

} game;
```

What would you need to do in `newGame` to set this up?

```
Game newGame (/* ... */) {
    // allocate memory for a game struct

    // set the turn number to start at 0

}
```

What about in `currentTurn`?

```
int currentTurn (Game game) {
    return game->turnNumber;
}
```

What about in playMove?

```
void playMove(Game game, playerMove move) {
    // if the player played an END_TURN move...

    // their turn has ended, so increase the turn number.
}
```

What would you need to add to also keep track of the current player?

Your tutor will discuss some/all of the following topics during your tutorial/lab this week. Make sure that you properly understand the following, if you have problems, please ask questions during your tutorial/lab or go to a help session.

- Identify values (in plain English) you need to store in the structure _game in Game.c
- How will you create a linked list of cards? (we discussed this in the lecture)
- How many linked list of cards do you need?
- How can you move cards (without copying or creating new cards) between linked list of cards?
- Discuss possible **operations (helper functions)** you may need on a linked list of cards for the assignment. For example: draw a card from the deck, place card on discard pile, remove a card from a hand, etc.
- If possible, try to identify **operations (helper functions)** that could be **reused**. You may need to carry our multiple operations to achieve your goal. For example, drawing a card from the deck may need two operations: removing a (which?) card from the deck (linked list of cards) and adding a card to the required hand (linked list of cards).
- Discuss (in plain English, not c code) how will you implement "`void` **`playMove`**`(Game game, playerMove move);`"
  - a move when a player plays a card with value 2.
  - a move when a player plays a card with value A.
  - a move when a player plays a card with value D and change color.
- Discuss (in plain English, not c code) how will you implement "`int` **`isValidMove`**`(Game game, playerMove move);`"

COMP1511 18s2: Introduction to Programming is brought to you by
the School of Computer Science and Engineering at the University of New South Wales, Sydney.
For all enquiries, please email the class account at cs1511@cse.unsw.edu.au
CRICOS Provider 00098G