
How to Write a Simple Random Tester

Building a Simple Random Tester

1. Identify the interface to test
 - Is it a file interface?
 - Network interface?
 - Calls to a function?
2. Write code to generate random inputs
 - What values is the code expected to handle?
 - Are all of these values interesting?
3. Write code to check behavior on random inputs
 - How can you tell if it worked?

Recipe for Refining a Random Tester

1. Gather code coverage
 - Is everything interesting being covered?
 - Is important code not covered?
2. Adjust the code to generate inputs
 - Try to “stay random” but shift the probability space
 - Augment random with fixed inputs of interest
3. Break the code and see if your tests detect the problem
 - If not, why not?
4. Improve your oracle code until all problems that should be caught *are* caught
5. Repeat until coverage and “fake bugs” both show the testing is rock solid

```
/cygdrive/c/Documents and Settings/Alex/Desktop/cs362class/dominion
File Edit Options Buffers Tools C Help
int drawCard(int player, struct gameState *state)
{
    int count;
    int deckCounter;
    if (state->deckCount[player] <= 0){//Deck is empty
        int i;
        for (i = 0; i < state->discardCount[player];i++){
            state->deck[player][i] = state->discard[player][i];
            state->discard[player][i] = -1;
        }

        state->deckCount[player] = state->discardCount[player];
        state->discardCount[player] = 0;//Reset discard

        shuffle(player, state);//Shuffle the deck up and make it so that we can draw

        if (DEBUG){//Debug statements
            printf("Deck count now: %d\n", state->deckCount[player]);
        }

        state->discardCount[player] = 0;

        count = state->handCount[player];//Get current player's hand count

        if (DEBUG){//Debug statements
            printf("Current hand count: %d\n", count);
        }

        deckCounter = state->deckCount[player];//Create a holder for the deck count

        if (deckCounter == 0)
            return -1;

        state->hand[player][count] = state->deck[player][deckCounter - 1];//Add card to hand
        state->deckCount[player]--;
        state->handCount[player]++;//Increment hand count
    }

    else{
        int count = state->handCount[player];//Get current hand count for player
        int deckCounter;
        if (DEBUG){//Debug statements
            printf("Current hand count: %d\n", count);
        }

        deckCounter = state->deckCount[player];//Create holder for the deck count
        state->hand[player][count] = state->deck[player][deckCounter - 1];//Add card to the hand
        state->deckCount[player]--;
        state->handCount[player]++;//Increment hand count
    }

    return 0;
}
```

-UU-:**--F1 dominion.c 38% L548 (C/1 Abbrev)-----

What code
are we
testing?

```
int drawCard(
```

```
    int player,
```

```
    struct gameState *state
```

```
);
```

**What
inputs
does it
take?**

```
int drawCard(
```

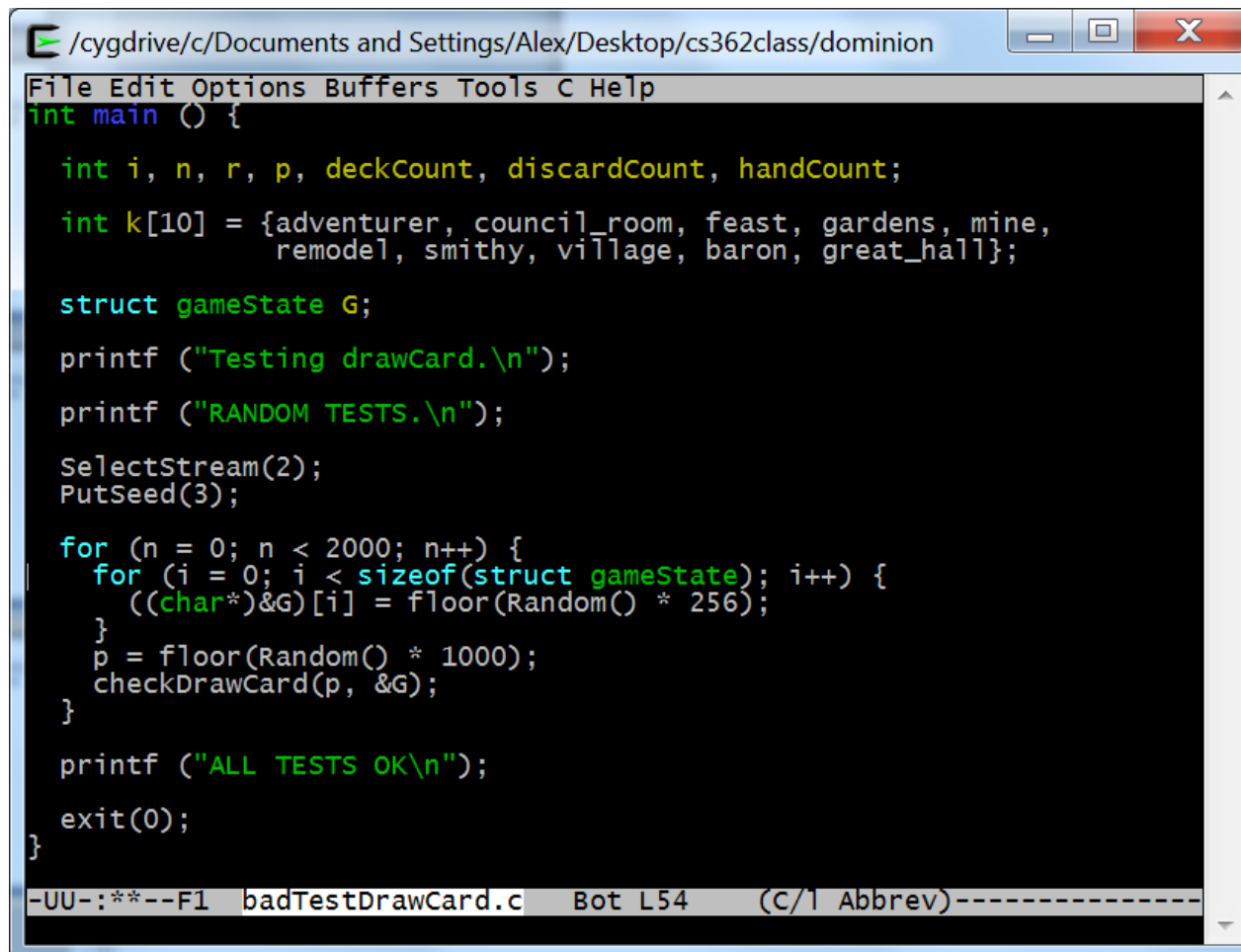
```
    int player,
```

```
    struct gameState *state
```

```
);
```

**Can we randomly
generate these
inputs?**

Yes...



```
File Edit Options Buffers Tools C Help
int main () {
    int i, n, r, p, deckCount, discardCount, handCount;
    int k[10] = {adventurer, council_room, feast, gardens, mine,
                remodel, smithy, village, baron, great_hall};
    struct gameState G;
    printf ("Testing drawCard.\n");
    printf ("RANDOM TESTS.\n");
    SelectStream(2);
    PutSeed(3);
    for (n = 0; n < 2000; n++) {
        for (i = 0; i < sizeof(struct gameState); i++) {
            ((char*)&G)[i] = floor(Random() * 256);
        }
        p = floor(Random() * 1000);
        checkDrawCard(p, &G);
    }
    printf ("ALL TESTS OK\n");
    exit(0);
}
```

-UU-:**--F1 badTestDrawCard.c Bot L54 (C/I Abbrev)-----

This code is generating random tests:

- 1. Create a gameState G filled with random bytes**
- 2. Choose a number of players randomly**
- 3. Call a function to test drawCard with these inputs**

```
/cygdrive/c/Documents and Settings/Alex/Desktop/cs362class/dominion
File Edit Options Buffers Tools C Help
int main () {
    int i, n, r, p, deckCount, discardCount, handCount;
    int k[10] = {adventurer, council_room, feast, gardens, mine,
                remodel, smithy, village, baron, great_hall};

    struct gameState G;
    printf ("Testing drawCard.\n");
    printf ("RANDOM TESTS.\n");
    SelectStream(2);
    PutSeed(3);

    for (n = 0; n < 2000; n++) {
        for (i = 0; i < sizeof(struct gameState); i++) {
            ((char*)&G)[i] = floor(Random() * 256);
        }
        p = floor(Random() * 1000);
        checkDrawCard(p, &G);
    }

    printf ("ALL TESTS OK\n");
    exit(0);
}
-UU-: **--F1 badTestDrawCard.c Bot L54 (C/I Abbrev)-----
```

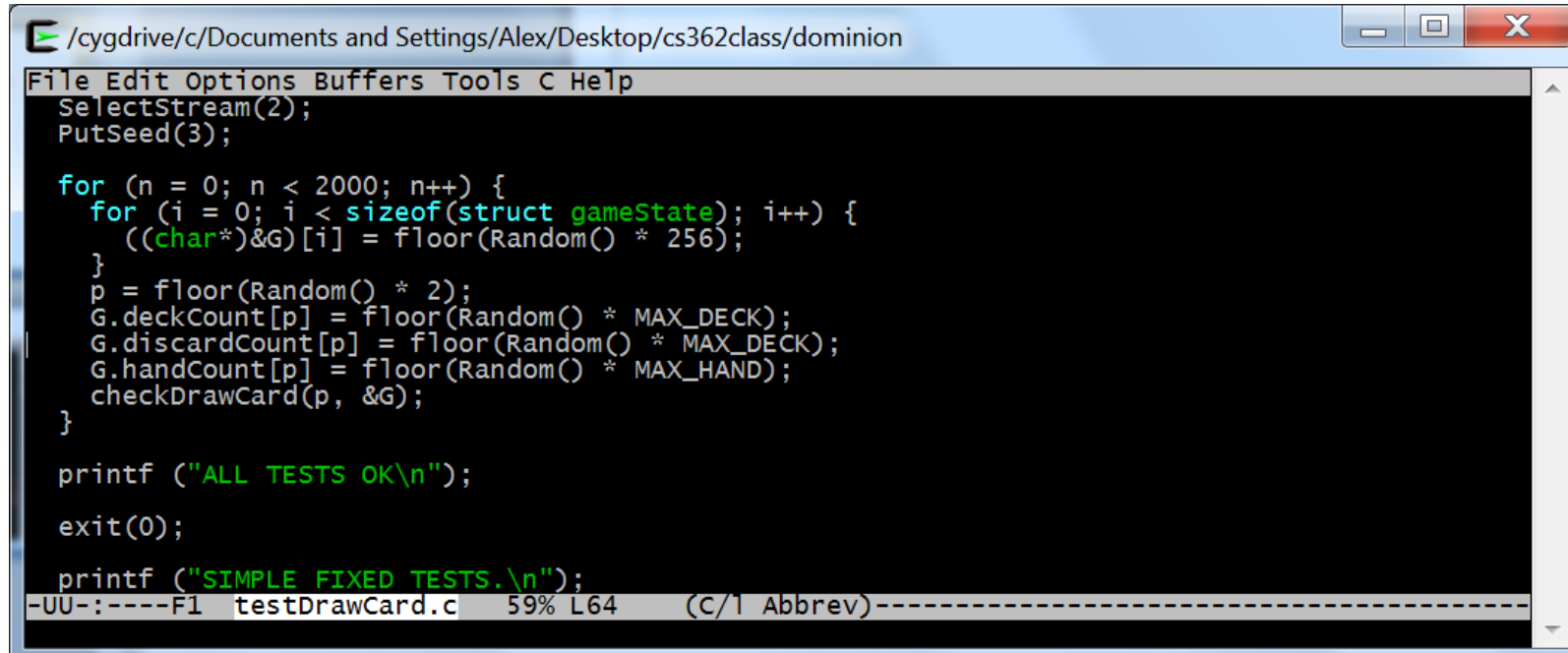
What happens
when we run
this tester?

```
/cygdrive/c/Documents and Settings/Alex/Desktop/cs362class/dominion
Alex@groce /cygdrive/c/Documents and Settings/Alex/Desktop/cs362class/dominion
$ ./badTestDrawCard.exe
Testing drawCard.
RANDOM TESTS.
Segmentation fault (core dumped)
Alex@groce /cygdrive/c/Documents and Settings/Alex/Desktop/cs362class/dominion
$
```


Pure Random Seldom Works!

- **Need to think about preconditions**

- drawCard expects
 - a valid number of players
 - a somewhat “sane” gameState
- Can we generate that?



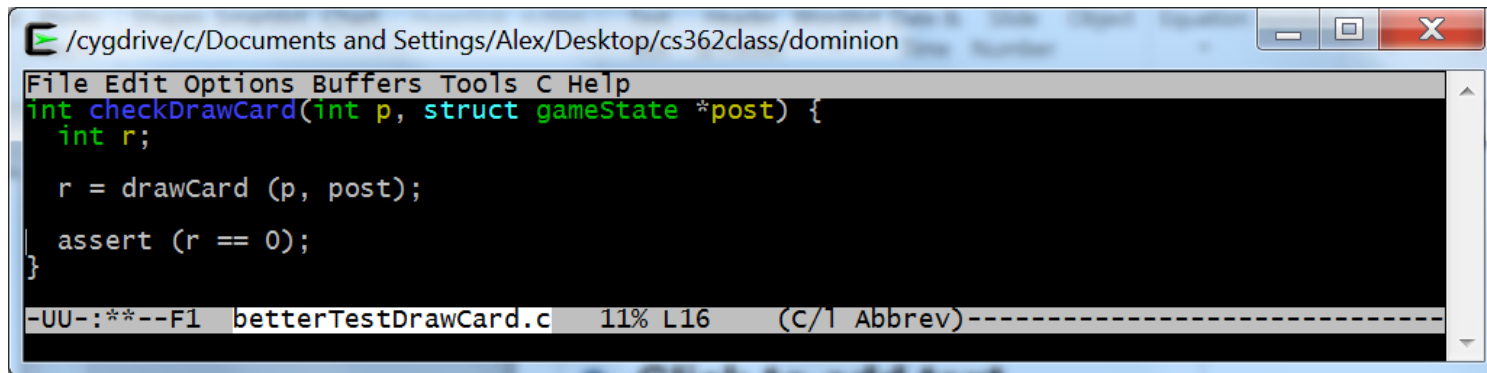
```
File Edit Options Buffers Tools C Help
SelectStream(2);
PutSeed(3);

for (n = 0; n < 2000; n++) {
    for (i = 0; i < sizeof(struct gameState); i++) {
        ((char*)&G)[i] = floor(Random() * 256);
    }
    p = floor(Random() * 2);
    G.deckCount[p] = floor(Random() * MAX_DECK);
    G.discardCount[p] = floor(Random() * MAX_DECK);
    G.handCount[p] = floor(Random() * MAX_HAND);
    checkDrawCard(p, &G);
}

printf ("ALL TESTS OK\n");
exit(0);

printf ("SIMPLE FIXED TESTS.\n");
-UU-:----F1 testDrawCard.c 59% L64 (C/1 Abbrev)-----
```

Check Your Test Oracle



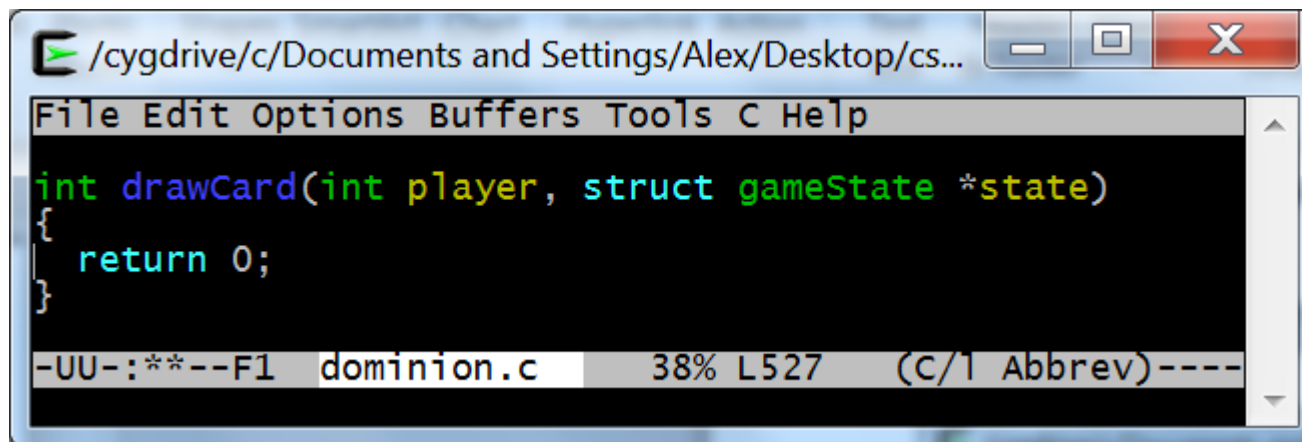
A screenshot of a code editor window titled `/cygdrive/c/Documents and Settings/Alex/Desktop/cs362class/dominion`. The editor displays the implementation of the `checkDrawCard` function in `betterTestDrawCard.c`. The code is as follows:

```
File Edit Options Buffers Tools C Help
int checkDrawCard(int p, struct gameState *post) {
    int r;

    r = drawCard (p, post);

    assert (r == 0);
}
```

The status bar at the bottom shows: `-UU-:***-F1 betterTestDrawCard.c 11% L16 (C/1 Abbrev)-----`.

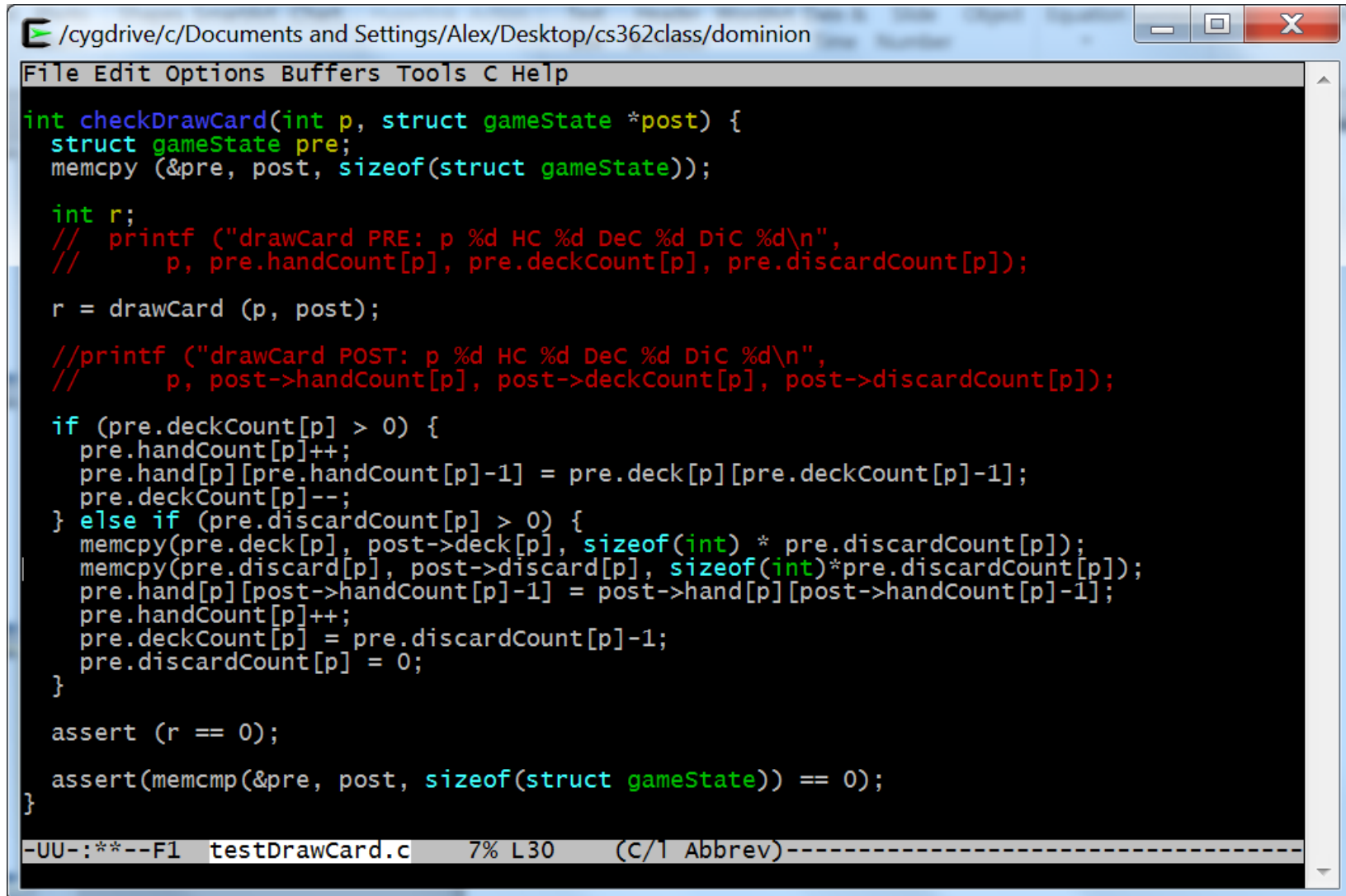


A screenshot of a code editor window titled `/cygdrive/c/Documents and Settings/Alex/Desktop/cs...`. The editor displays the implementation of the `drawCard` function in `dominion.c`. The code is as follows:

```
File Edit Options Buffers Tools C Help
int drawCard(int player, struct gameState *state)
{
    return 0;
}
```

The status bar at the bottom shows: `-UU-:***-F1 dominion.c 38% L527 (C/1 Abbrev)----`.

Revise Your Test Oracle



```

/cygdrive/c/Documents and Settings/Alex/Desktop/cs362class/dominion
File Edit Options Buffers Tools C Help

int checkDrawCard(int p, struct gameState *post) {
    struct gameState pre;
    memcpy (&pre, post, sizeof(struct gameState));

    int r;
    // printf ("drawCard PRE: p %d HC %d DeC %d DiC %d\n",
    //         p, pre.handCount[p], pre.deckCount[p], pre.discardCount[p]);

    r = drawCard (p, post);

    //printf ("drawCard POST: p %d HC %d DeC %d DiC %d\n",
    //        p, post->handCount[p], post->deckCount[p], post->discardCount[p]);

    if (pre.deckCount[p] > 0) {
        pre.handCount[p]++;
        pre.hand[p][pre.handCount[p]-1] = pre.deck[p][pre.deckCount[p]-1];
        pre.deckCount[p]--;
    } else if (pre.discardCount[p] > 0) {
        memcpy(pre.deck[p], post->deck[p], sizeof(int) * pre.discardCount[p]);
        memcpy(pre.discard[p], post->discard[p], sizeof(int)*pre.discardCount[p]);
        pre.hand[p][post->handCount[p]-1] = post->hand[p][post->handCount[p]-1];
        pre.handCount[p]++;
        pre.deckCount[p] = pre.discardCount[p]-1;
        pre.discardCount[p] = 0;
    }

    assert (r == 0);

    assert(memcmp(&pre, post, sizeof(struct gameState)) == 0);
}

-UU-:***-F1  testDrawCard.c  7% L30  (C/1 Abbrev)-----

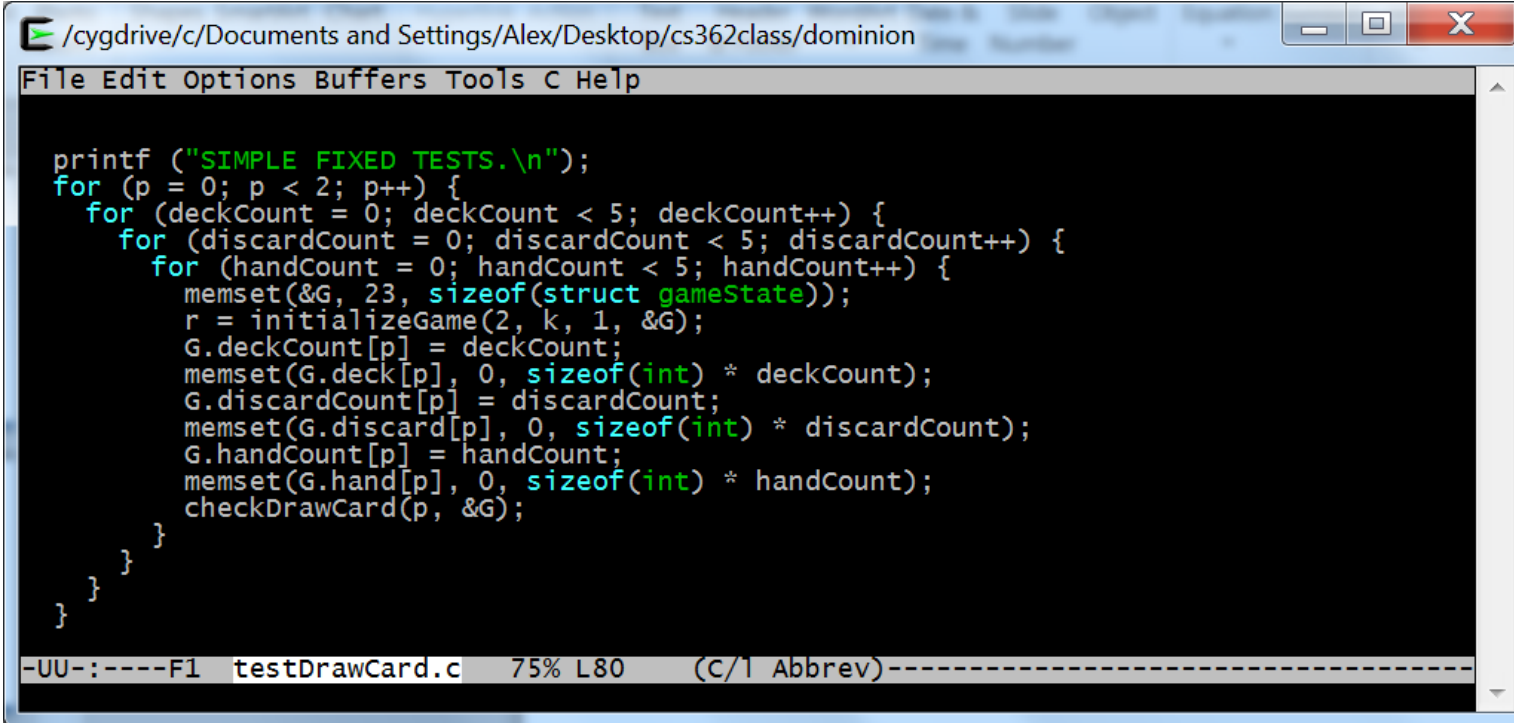
```

Check Coverage

```
/cygdrive/c/Documents and Settings/Alex/Desktop/cs362class/dominion

-: 524:
2000: 525: int drawCard(int player, struct gameState *state)
-: 526: {
-: 527:     int count;
2000: 528:     if (state->deckCount[player] <= 0){//Deck is empty
-: 529:
-: 530:         //Step 1 Shuffle the discard pile back into a deck
-: 531:         int i;
-: 532:         //Move discard to deck
811: 533:         for (i = 0; i < state->discardCount[player]; i++){
809: 534:             state->deck[player][i] = state->discard[player][i];
809: 535:             state->discard[player][i] = -1;
-: 536:         }
-: 537:
2: 538:         state->deckCount[player] = state->discardCount[player];
2: 539:         state->discardCount[player] = 0;//Reset discard
-: 540:
-: 541:         //Shuffle the deck
2: 542:         shuffle(player, state);//Shuffle the deck up and make it so that we can draw
-: 543:
-: 544:         if (DEBUG){//Debug statements
-: 545:             printf("Deck count now: %d\n", state->deckCount[player]);
-: 546:         }
-: 547:
2: 548:         state->discardCount[player] = 0;
-: 549:
-: 550:         //Step 2 Draw Card
2: 551:         count = state->handCount[player];//Get current player's hand count
-: 552:
-: 553:         if (DEBUG){//Debug statements
-: 554:             printf("Current hand count: %d\n", count);
-: 555:         }
-: 556:
2: 557:         deckCounter = state->deckCount[player];//Create a holder for the deck count
-: 558:
2: 559:         if (deckCounter == 0)
#####: 560:             return -1;
-: 561:
2: 562:         state->hand[player][count] = state->deck[player][deckCounter - 1];//Add card to h
and
2: 563:         state->deckCount[player]--;
2: 564:         state->handCount[player]++;//Increment hand count
-: 565:     }
-: 566:
-: 567:     else{
1998: 568:         int count = state->handCount[player];//Get current hand count for player
-: 569:         int deckCounter;
-: 570:         if (DEBUG){//Debug statements
-: 571:
-: 572:         }
```

Add Fixed Tests if Needed

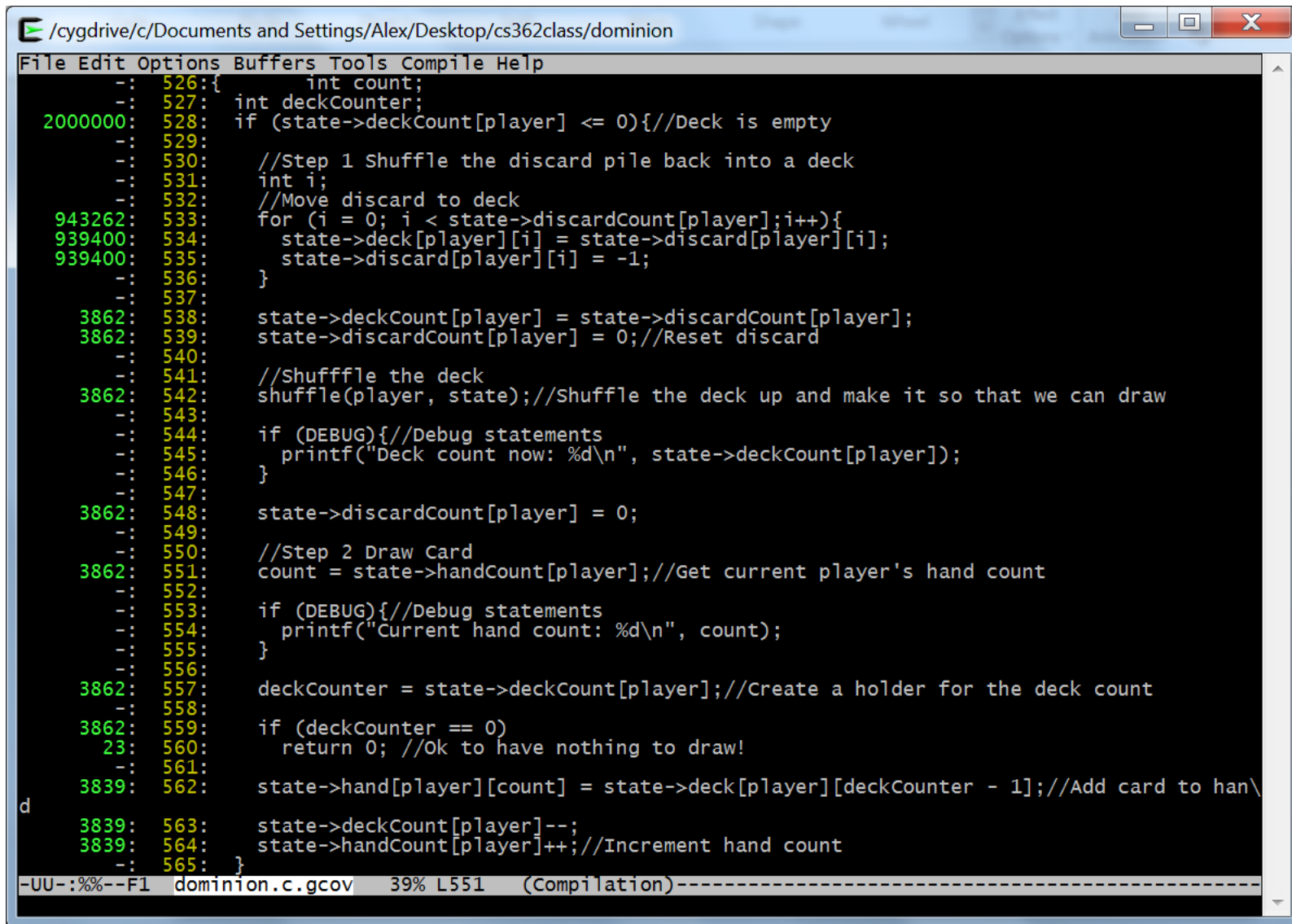
A screenshot of a code editor window. The title bar shows the file path: /cygdrive/c/Documents and Settings/Alex/Desktop/cs362class/dominion. The menu bar includes File, Edit, Options, Buffers, Tools, C, and Help. The code is written in C and uses syntax highlighting. It features three nested for loops to iterate over different game states (p, deckCount, discardCount, handCount) and calls various game functions like memset, initializeGame, and checkDrawCard. The status bar at the bottom indicates the file is testDrawCard.c, with 75% line coverage (L80) and a comment abbreviation (C/1 Abbrev).

```
printf ("SIMPLE FIXED TESTS.\n");
for (p = 0; p < 2; p++) {
    for (deckCount = 0; deckCount < 5; deckCount++) {
        for (discardCount = 0; discardCount < 5; discardCount++) {
            for (handCount = 0; handCount < 5; handCount++) {
                memset(&G, 23, sizeof(struct gameState));
                r = initializeGame(2, k, 1, &G);
                G.deckCount[p] = deckCount;
                memset(G.deck[p], 0, sizeof(int) * deckCount);
                G.discardCount[p] = discardCount;
                memset(G.discard[p], 0, sizeof(int) * discardCount);
                G.handCount[p] = handCount;
                memset(G.hand[p], 0, sizeof(int) * handCount);
                checkDrawCard(p, &G);
            }
        }
    }
}
```

-UU-:----F1 testDrawCard.c 75% L80 (C/1 Abbrev)-----

Final testDrawCard.c gets coverage of every line of drawCard, including the empty-deck and discard case (10 times). You can also get this result by going from 2,000 tests to 2,000,000 – which covers the empty case 23 times and only takes half an hour to run. You also get more coverage of everything else.

Don't Work Smarter, Just Work Harder?



```
File Edit Options Buffers Tools Compile Help
-: 526: { int count;
-: 527: int deckCounter;
2000000: 528: if (state->deckCount[player] <= 0){//Deck is empty
-: 529:
-: 530: //Step 1 Shuffle the discard pile back into a deck
-: 531: int i;
-: 532: //Move discard to deck
943262: 533: for (i = 0; i < state->discardCount[player];i++){
939400: 534: state->deck[player][i] = state->discard[player][i];
939400: 535: state->discard[player][i] = -1;
-: 536: }
-: 537:
3862: 538: state->deckCount[player] = state->discardCount[player];
3862: 539: state->discardCount[player] = 0;//Reset discard
-: 540:
-: 541: //Shuffle the deck
3862: 542: shuffle(player, state);//Shuffle the deck up and make it so that we can draw
-: 543:
-: 544: if (DEBUG){//Debug statements
-: 545: printf("Deck count now: %d\n", state->deckCount[player]);
-: 546: }
-: 547:
3862: 548: state->discardCount[player] = 0;
-: 549:
-: 550: //Step 2 Draw Card
3862: 551: count = state->handCount[player];//Get current player's hand count
-: 552:
-: 553: if (DEBUG){//Debug statements
-: 554: printf("Current hand count: %d\n", count);
-: 555: }
-: 556:
3862: 557: deckCounter = state->deckCount[player];//Create a holder for the deck count
-: 558:
3862: 559: if (deckCounter == 0)
23: 560: return 0; //Ok to have nothing to draw!
-: 561:
3839: 562: state->hand[player][count] = state->deck[player][deckCounter - 1];//Add card to han\
d
3839: 563: state->deckCount[player]--;
3839: 564: state->handCount[player]++;//Increment hand count
-: 565: }
```

-UU-:%%--F1 dominion.c.gcov 39% L551 (Compilation)-----