

Brian Laccone

CS 362

5/27/18

Assignment-5

Bug-Reports

Three bugs were found during the testing of my teammate's, Jeff Loung, dominion program. Both the bugs were found in the refactor adventurer and smithy card functions. Adventurer and Smithy were the only refactored cards that our programs had in common. All of the other tests unit and randomized tests were done for cards and functions that were untouched by my teammate.

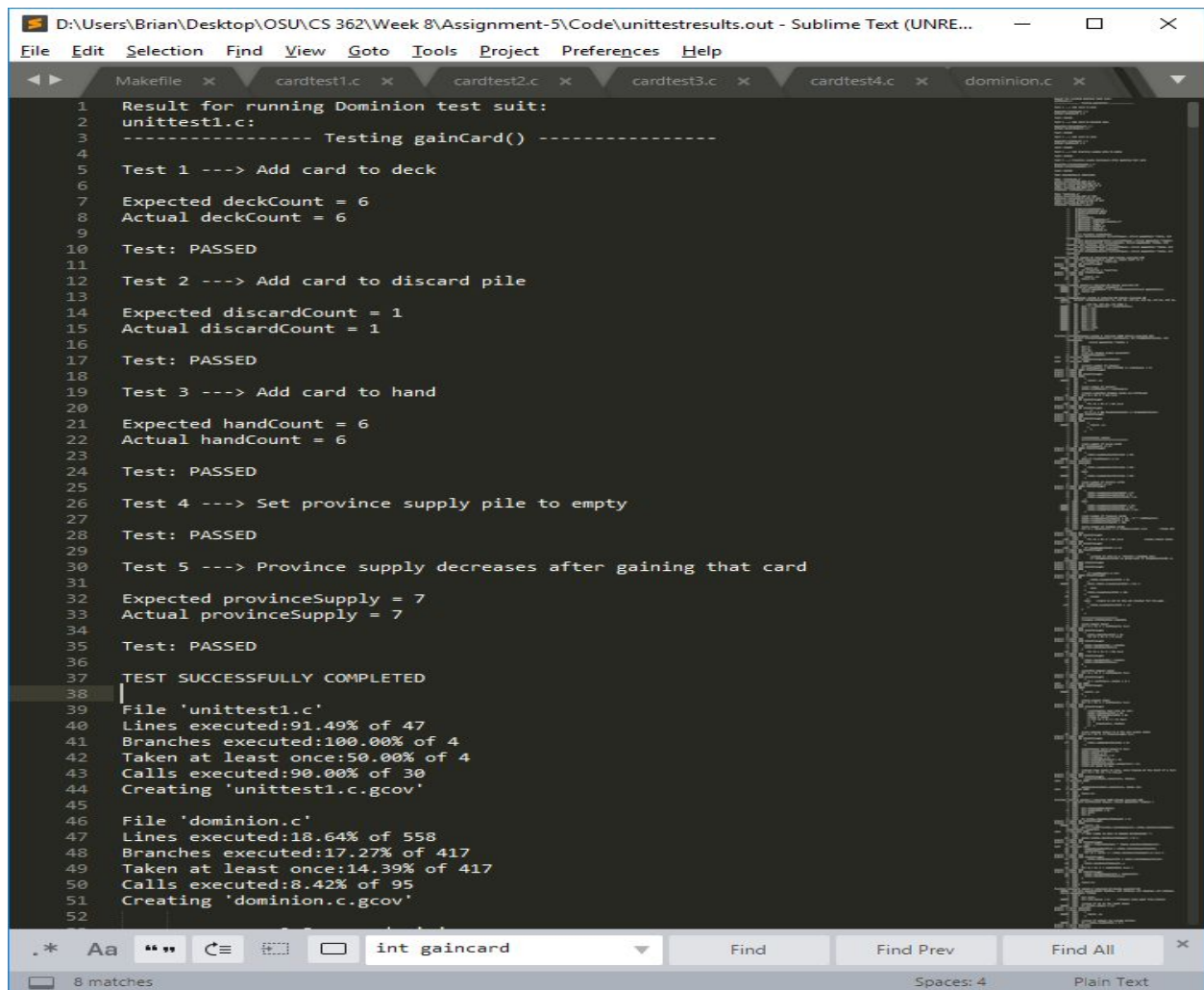
1-2. AdventurerCard() - Both cardtest1.c and randomtestadventurer.c discovered that the hand counts before and after this function was called did not match up. randomtestadventurer.c also discovered that the total card count was wrong as well. The hand count was always 1 less more than it should have been. This suggests that either adventurer was not being discarded, the function was drawing more than two cards, or the way the cards are being counted is wrong. I first discovered the hand count bug when I tested cardtest1.c. The hand count was always 7 instead of 6. I first discovered the total card count bug when I ran the randomtestadventurer.c test. The count was off everytime no matter how many iterations I ran.

3. SmithyCard() - The third bug was discovered in both cardtest2.c and randomtest2.c. These tests discovered that the hand counts were wrong before and after SmithyCard was called. The hand count was always 1 more than what the count should have been. I first discovered this bug during the cardtest2.c and I confirmed it again when the randomtest2.c test ran. The randomtest2.c was about a 99% failure rate. The reason it was not 100 is because of the random values for the deck count. Occasionally the deck would only have three cards in it so the hand count would have been correct then.

Test Report

unittest1.c

My first unit test covered the function `gainCard()`. The test received 100% statement and branch coverage for this function. Unfortunately my overall coverage of `dominion.c` was only 18.64% with 17.27% branch coverage. I could have feed the function several different states to try and test what it would do with better defined states instead of just the same state every call to it. I made sure that I hit every branch by feeding the function supply counts that were empty as well as all three “toFlag” branch options. There were no problem with this function as it was not refactored by my teammate and passed all the tests.



```
1 Result for running Dominion test suit:
2 unittest1.c:
3 ----- Testing gainCard() -----
4
5 Test 1 ---> Add card to deck
6
7 Expected deckCount = 6
8 Actual deckCount = 6
9
10 Test: PASSED
11
12 Test 2 ---> Add card to discard pile
13
14 Expected discardCount = 1
15 Actual discardCount = 1
16
17 Test: PASSED
18
19 Test 3 ---> Add card to hand
20
21 Expected handCount = 6
22 Actual handCount = 6
23
24 Test: PASSED
25
26 Test 4 ---> Set province supply pile to empty
27
28 Test: PASSED
29
30 Test 5 ---> Province supply decreases after gaining that card
31
32 Expected provinceSupply = 7
33 Actual provinceSupply = 7
34
35 Test: PASSED
36
37 TEST SUCCESSFULLY COMPLETED
38
39 File 'unittest1.c'
40 Lines executed:91.49% of 47
41 Branches executed:100.00% of 4
42 Taken at least once:50.00% of 4
43 Calls executed:90.00% of 30
44 Creating 'unittest1.c.gcov'
45
46 File 'dominion.c'
47 Lines executed:18.64% of 558
48 Branches executed:17.27% of 417
49 Taken at least once:14.39% of 417
50 Calls executed:8.42% of 95
51 Creating 'dominion.c.gcov'
52
```

D:\Users\Brian\Desktop\OSU\CS 362\Week 4\Assignment 3\unittestresults.out - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

dominion.c x mgs.c x Makefile x unittestresults.out x

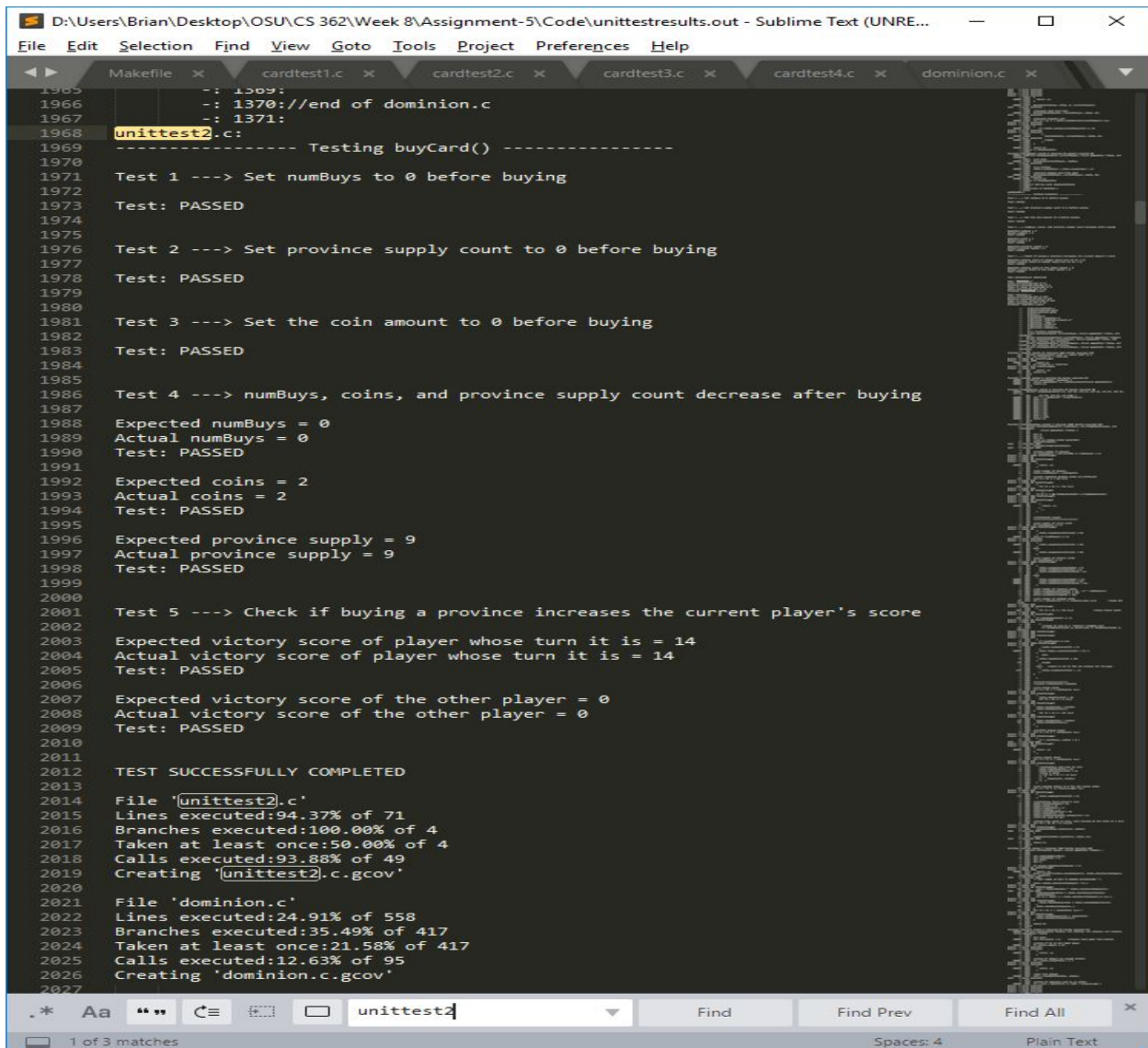
```
1926 -: 1350:
1927 function gainCard called 5 returned 100% blocks executed 100%
1928 ..... 5: 1351: int gainCard(int supplyPos, struct gameState *state, int toFlag, int player)
1929 ..... -: 1352: {
1930 ..... -: 1353: //Note: supplyPos is enum of choosen card
1931 ..... -: 1354:
1932 ..... -: 1355: //check if supply pile is empty (0) or card is not used in game (-1)
1933 ..... 5: 1356: if ( supplyCount(supplyPos, state) < 1 )
1934 call 0 returned 100%
1935 branch 1 taken 20% (fallthrough)
1936 branch 2 taken 80%
1937 ..... -: 1357: {
1938 ..... 1: 1358: return -1;
1939 ..... -: 1359: }
1940 ..... -: 1360:
1941 ..... -: 1361: //added card for [whoseTurn] current player:
1942 ..... -: 1362: // toFlag = 0 : add to discard
1943 ..... -: 1363: // toFlag = 1 : add to deck
1944 ..... -: 1364: // toFlag = 2 : add to hand
1945 ..... -: 1365:
1946 ..... 4: 1366: if (toFlag == 1)
1947 branch 0 taken 25% (fallthrough)
1948 branch 1 taken 75%
1949 ..... -: 1367: {
1950 ..... 1: 1368: state->deck[player][state->deckCount[player]] = supplyPos;
1951 ..... 1: 1369: state->deckCount[player]++;
1952 ..... -: 1370: }
1953 ..... 3: 1371: else if (toFlag == 2)
1954 branch 0 taken 33% (fallthrough)
1955 branch 1 taken 67%
1956 ..... -: 1372: {
1957 ..... 1: 1373: state->hand[player][state->handCount[player]] = supplyPos;
1958 ..... 1: 1374: state->handCount[player]++;
1959 ..... -: 1375: }
1960 ..... -: 1376: else
1961 ..... -: 1377: {
1962 ..... 2: 1378: state->discard[player][state->discardCount[player]] = supplyPos;
1963 ..... 2: 1379: state->discardCount[player]++;
1964 ..... -: 1380: }
1965 ..... -: 1381:
1966 ..... -: 1382: //decrease number in supply pile
1967 ..... 4: 1383: state->supplyCount[supplyPos]--;
1968 ..... -: 1384:
1969 ..... 4: 1385: return 0;
1970 ..... -: 1386: }
1971 ..... -: 1387:
1972 function updateCoins called 1 returned 100% blocks executed 82%
1973 ..... 1: 1388: int updateCoins(int player, struct gameState *state, int bonus)
```

* Aa " " " int gainCard Find Find Prev Find All x

8 matches Spaces: 4 Plain Text

unittest2.c

My unittest2.c file contained the unit test for the buyCard() function. The file says that I had returned 100% and had 100% blocks executed. I had 100% branch coverage, which I find interesting because I didn't hit any of the DEBUG branches. DEBUG was meant to be a visual representation of the sequence of code in the buyCard() function. I didn't hit the print command in any of the if (DEBUG) {} statements. My guess is that gcov has a built in feature to skip DEBUG branches. My statement coverage of the whole dominion.c file was 24.91% and my branch coverage was 35.49%. There were no problem with this function as it was not refactored by my teammate and passed all the tests.



```
D:\Users\Brian\Desktop\OSU\CS 362\Week 8\Assignment-5\Code\unittestresults.out - Sublime Text (UNRE...
File Edit Selection Find View Goto Tools Project Preferences Help
Makefile x cardtest1.c x cardtest2.c x cardtest3.c x cardtest4.c x dominion.c x
1965 --: 1369:
1966 --: 1370://end of dominion.c
1967 --: 1371:
1968 unittest2.c:
1969 ----- Testing buyCard() -----
1970
1971 Test 1 ---> Set numBuys to 0 before buying
1972
1973 Test: PASSED
1974
1975
1976 Test 2 ---> Set province supply count to 0 before buying
1977
1978 Test: PASSED
1979
1980
1981 Test 3 ---> Set the coin amount to 0 before buying
1982
1983 Test: PASSED
1984
1985
1986 Test 4 ---> numBuys, coins, and province supply count decrease after buying
1987
1988 Expected numBuys = 0
1989 Actual numBuys = 0
1990 Test: PASSED
1991
1992 Expected coins = 2
1993 Actual coins = 2
1994 Test: PASSED
1995
1996 Expected province supply = 9
1997 Actual province supply = 9
1998 Test: PASSED
1999
2000
2001 Test 5 ---> Check if buying a province increases the current player's score
2002
2003 Expected victory score of player whose turn it is = 14
2004 Actual victory score of player whose turn it is = 14
2005 Test: PASSED
2006
2007 Expected victory score of the other player = 0
2008 Actual victory score of the other player = 0
2009 Test: PASSED
2010
2011
2012 TEST SUCCESSFULLY COMPLETED
2013
2014 File 'unittest2.c'
2015 Lines executed:94.37% of 71
2016 Branches executed:100.00% of 4
2017 Taken at least once:50.00% of 4
2018 Calls executed:93.88% of 49
2019 Creating 'unittest2.c.gcov'
2020
2021 File 'dominion.c'
2022 Lines executed:24.91% of 558
2023 Branches executed:35.49% of 417
2024 Taken at least once:21.58% of 417
2025 Calls executed:12.63% of 95
2026 Creating 'dominion.c.gcov'
2027
.* Aa " " C E 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000
1 of 3 matches Spaces: 4 Plain Text
```


D:\Users\Brian\Desktop\OSU\CS 362\Week 4\Assignment 3\Code\unittestresults.out - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

dominion.c x unittestresults.out x

```
2433 -: 271:
2434 function buyCard called 5 returned 100% blocks executed 100%
2435 .....5: 272: int buyCard(int supplyPos, struct gameState *state){
2436 ..... -: 273: int who;
2437 ..... -: 274: if (DEBUG){
2438 ..... -: 275: printf("Entering buyCard...\n");
2439 ..... -: 276: }
2440 ..... -: 277:
2441 ..... -: 278: // I don't know what to do about the phase thing.
2442 ..... -: 279:
2443 .....5: 280: who = state->whoseTurn;
2444 ..... -: 281:
2445 .....5: 282: if (state->numBuys < 1){
2446 branch 0 taken 20% (fallthrough)
2447 branch 1 taken 80%
2448 ..... -: 283: if (DEBUG)
2449 ..... -: 284: printf("You do not have any buys left\n");
2450 .....1: 285: return -1;
2451 .....4: 286: } else if (supplyCount(supplyPos, state) < 1){
2452 call 0 returned 100%
2453 branch 1 taken 25% (fallthrough)
2454 branch 2 taken 75%
2455 ..... -: 287: if (DEBUG)
2456 ..... -: 288: printf("There are not any of that type of card left\n");
2457 .....1: 289: return -1;
2458 .....3: 290: } else if (state->coins < getCost(supplyPos)){
2459 call 0 returned 100%
2460 branch 1 taken 33% (fallthrough)
2461 branch 2 taken 67%
2462 ..... -: 291: if (DEBUG)
2463 ..... -: 292: printf("You do not have enough money to buy that. You have %d coins.\n",
state->coins);
2464 .....1: 293: return -1;
2465 ..... -: 294: } else {
2466 .....2: 295: state->phase = 1;
2467 ..... -: 296: //state->supplyCount[supplyPos]--;
2468 .....2: 297: gainCard(supplyPos, state, 0, who); //card goes in discard, this might be
wrong.. (2 means goes into hand, 0 goes into discard)
2469 call 0 returned 100%
2470 ..... -: 298:
2471 .....2: 299: state->coins = (state->coins) - (getCost(supplyPos));
2472 call 0 returned 100%
2473 .....2: 300: state->numBuys--;
2474 ..... -: 301: if (DEBUG)
2475 ..... -: 302: printf("You bought card number %d for %d coins. You now have %d buys and %d
coins.\n", supplyPos, getCost(supplyPos), state->numBuys, state->coins);
2476 ..... -: 303: }
2477 ..... -: 304:
2478 ..... -: 305: //state->discard[who][state->discardCount[who]] = supplyPos;
2479 ..... -: 306: //state->discardCount[who]++;
2480 ..... -: 307:
2481 .....2: 308: return 0;
2482 ..... -: 309: }
2483 ..... -: 310:
2484 function numHandCards called 0 returned 0% blocks executed 0%
```

* Aa 8 matches int buycard Find Find Prev Find All x

Spaces: 4 Plain Text

unittest3.c

My unittest3.c file contains the tests for the isGameOver() function. This test got 100% statement coverage and 100% branch coverage for the isGameOver() function. Luckily, this was a very simple function and only took me 3 calls of it to cover 100%. My coverage off dominion.c was 17.74% statement and 17.75% branch coverage. There were no problem with this function as it was not refactored by my teammate and passed all the tests.

The screenshot shows a Sublime Text editor window with the file path `D:\Users\Brian\Desktop\OSU\CS 362\Week 8\Assignment-5\Code\unittestresults.out - Sublime Text (UNRE...`. The editor has several tabs open: `Makefile`, `cardtest1.c`, `cardtest2.c`, `cardtest3.c`, `cardtest4.c`, and `dominion.c`. The main content area displays the output of a C program, which is the file `unittest3.c`. The output shows three tests passing and the program completing successfully. The output also includes coverage statistics for `unittest3.c` and `dominion.c`.

```

3942      -: 13/1:
3943  unittest3.c:
3944  ----- Testing isGameOver() -----
3945
3946  Test 1 ---> Province supply pile set to 0
3947
3948  Test: PASSED
3949
3950
3951  Test 2 ---> Set three supply piles to 0
3952
3953  Test: PASSED
3954
3955  Test 3 ---> Province and all supply piles are not empty
3956
3957  Test: PASSED
3958
3959  TEST SUCCESSFULLY COMPLETED
3960
3961  File 'unittest3.c'
3962  Lines executed:87.88% of 33
3963  Branches executed:100.00% of 4
3964  Taken at least once:50.00% of 4
3965  Calls executed:82.35% of 17
3966  Creating 'unittest3.c.gcov'
3967
3968  File 'dominion.c'
3969  Lines executed:17.74% of 558
3970  Branches executed:17.75% of 417
3971  Taken at least once:14.87% of 417
3972  Calls executed:7.37% of 95
3973  Creating 'dominion.c.gcov'
3974

```

The bottom status bar shows the search results for the text `unittest3`, indicating 3 matches. The status bar also shows the current file is `unittest3`, the search is for `unittest3`, and the text is in `Plain Text` format.

D:\Users\Brian\Desktop\OSU\CS 362\Week 4\Assignment 3\Code\unittestresults.out - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

dominion.c x unittestresults.out x

```
4583     -: 388:}
4584     -: 389:
4585 function isGameOver called 3 returned 100% blocks executed 100%
4586 .....3: 390:int isGameOver(struct gameState *state){
4587 ..... -: 391: int i;
4588 ..... -: 392: int j;
4589 ..... -: 393:
4590 ..... -: 394: //if stack of Province cards is empty, the game ends
4591 .....3: 395: if (state->supplyCount[province] == 0)
4592 branch 0 taken 33% (fallthrough)
4593 branch 1 taken 67%
4594 ..... -: 396: {
4595 .....1: 397:     return 1;
4596 ..... -: 398: }
4597 ..... -: 399:
4598 ..... -: 400: //if three supply pile are at 0, the game ends
4599 .....2: 401: j = 0;
4600 .....52: 402: for (i = 0; i < 25; i++)
4601 branch 0 taken 96%
4602 branch 1 taken 4% (fallthrough)
4603 ..... -: 403: {
4604 .....50: 404:     if (state->supplyCount[i] == 0)
4605 branch 0 taken 6% (fallthrough)
4606 branch 1 taken 94%
4607 ..... -: 405:     {
4608 .....3: 406:         j++;
4609 ..... -: 407:     }
4610 ..... -: 408: }
4611 .....2: 409: if (j >= 3)
4612 branch 0 taken 50% (fallthrough)
4613 branch 1 taken 50%
4614 ..... -: 410: {
4615 .....1: 411:     return 1;
4616 ..... -: 412: }
4617 ..... -: 413:
4618 .....1: 414: return 0;
4619 ..... -: 415: }
4620 ..... -: 416:
4621 function scoreFor called 0 returned 0% blocks executed 0%
```

* Aa " " ☰ ☲ ☳ ☴ ☵ ☶ ☷ int isgame Find Find Prev Find All x

8 matches Spaces: 4 Plain Text

unittest4.c

My unittest4.c file was created to test the shuffle() file. The test received 100% statement and 100% branch coverage for the function. Unfortunately, it only received a 16.31% statement and 15.83% branch coverage for the whole dominion.c file. There were no problem with this function as it was not refactored by my teammate and passed all the tests.

D:\Users\Brian\Desktop\OSU\CS 362\Week 8\Assignment-5\Code\unitttestresults.out - Sublime Text (UNREGI...

File Edit Selection Find View Goto Tools Project Preferences Help

Makefile x cardtest1.c x cardtest2.c x cardtest3.c x cardtest4.c x dominion.c x

```

5889 -: 1371:
5890 unitttest4.c:
5891 ----- Testing shuffle() -----
5892
5893 Test 1 ---> Set player's deckCount to 0
5894
5895 Test: PASSED
5896
5897
5898 Test 2 ---> Make sure the deck is actually shuffled
5899
5900 Test: PASSED
5901
5902
5903 Test 3 ---> Test if the player has the same amount of cards in the deck before and
5904 after shuffling
5905
5906 Expected deckCount = 20
5907 Actual deckCount = 20
5908
5909 Test: PASSED
5910
5911 TEST SUCCESSFULLY COMPLETED
5912
5913 File 'unitttest4.c'
5914 Lines executed:87.18% of 39
5915 Branches executed:100.00% of 6
5916 Taken at least once:50.00% of 6
5917 Calls executed:81.82% of 22
5918 Creating 'unitttest4.c.gcov'
5919
5920 File 'dominion.c'
5921 Lines executed:16.31% of 558
5922 Branches executed:15.83% of 417
5923 Taken at least once:13.43% of 417
5924 Calls executed:7.37% of 95
5925 Creating 'dominion.c.gcov'
5926

```

.* Aa " " ☰ ☐ unitttest4 ▼ Find Find Prev Find All x

☐ 1 of 3 matches Spaces: 4 Plain Text

D:\Users\Brian\Desktop\OSU\CS 362\Week 4\Assignment 3\Code\unittestresults.out - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

dominion.c x unittestresults.out x

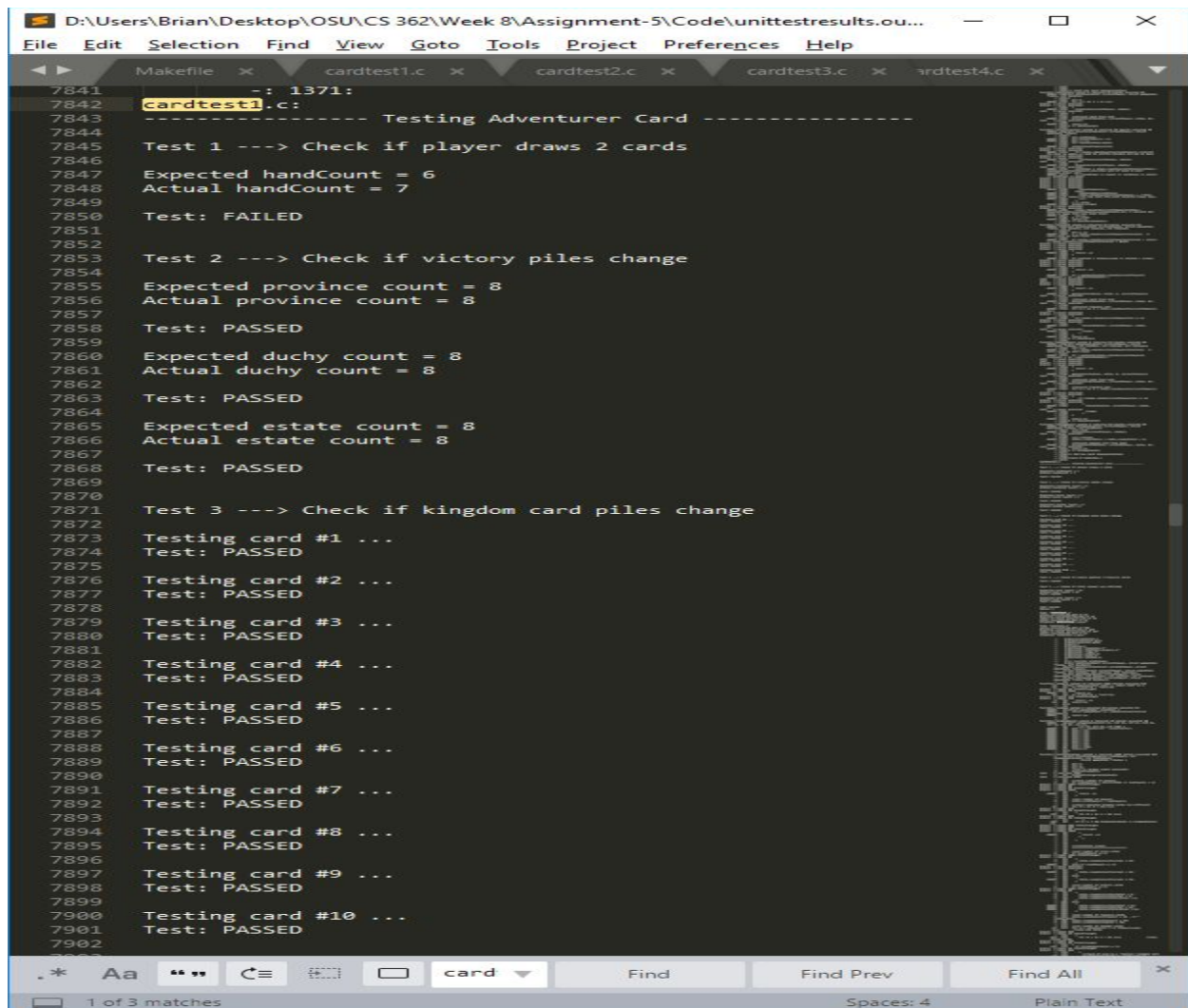
```
6329 function shuffle called 5 returned 100% blocks executed 100%
6330 .....5: 201: int shuffle(int player, struct gameState *state) {
6331 .....: 202:
6332 .....: 203:
6333 .....: 204:     int newDeck[MAX_DECK];
6334 .....5: 205:     int newDeckPos = 0;
6335 .....: 206:     int card;
6336 .....: 207:     int i;
6337 .....: 208:
6338 .....5: 209:     if (state->deckCount[player] < 1)
6339 branch 0 taken 20% (fallthrough)
6340 branch 1 taken 80%
6341 .....1: 210:     return -1;
6342 .....4: 211:     qsort((void*)(state->deck[player]), state->deckCount[player], sizeof(int),
        compare);
6343 call .....0 returned 100%
6344 .....: 212:     /* SORT CARDS IN DECK TO ENSURE DETERMINISM! */
6345 .....: 213:
6346 .....68: 214:     while (state->deckCount[player] > 0) {
6347 branch 0 taken 94%
6348 branch 1 taken 6% (fallthrough)
6349 .....60: 215:     card = floor(Random() * state->deckCount[player]);
6350 call .....0 returned 100%
6351 .....60: 216:     newDeck[newDeckPos] = state->deck[player][card];
6352 .....60: 217:     newDeckPos++;
6353 .....295: 218:     for (i = card; i < state->deckCount[player]-1; i++) {
6354 branch 0 taken 80%
6355 branch 1 taken 20% (fallthrough)
6356 .....235: 219:     state->deck[player][i] = state->deck[player][i+1];
6357 .....: 220:     }
6358 .....60: 221:     state->deckCount[player]--;
6359 .....: 222:     }
6360 .....64: 223:     for (i = 0; i < newDeckPos; i++) {
6361 branch 0 taken 94%
6362 branch 1 taken 6% (fallthrough)
6363 .....60: 224:     state->deck[player][i] = newDeck[i];
6364 .....60: 225:     state->deckCount[player]++;
6365 .....: 226:     }
6366 .....: 227:
6367 .....4: 228:     return 0;
6368 .....: 229: }
6369 .....: 230:
6370 function playCard called 0 returned 0% blocks executed 0%
6371 ##### 231: int playCard(int handPos, int choice1, int choice2, struct gameState
```

* Aa " " ☰ ☲ ☳ ☴ ☵ ☶ ☷ int shuffle Find Find Prev Find All x

8 matches Spaces: 4 Plain Text

cardtest1.c

My cardtest1.c was created to test the adventurer card. The test only had 73% statement coverage and missed a few branches. The first branch that it missed was the shuffle call. I didn't give the player and empty deck to test whether the shuffle call worked as intended. The next branch I missed contained a problem where the cards that were drawn were only treasure cards. This was the test's biggest coverage failure. It didn't randomize the cards in the deck nor did it have decks where no treasure cards existed. This test lacked boundary tests as well. Next time I would definitely put certain cards in the deck and test it that way. My dominion statement coverage was 19.89% and the branch coverage was 23.26%. There was a failure in one test. The hand count was one more than it should have been.



```
7841 -: 1371:
7842 cardtest1.c:
7843 ----- Testing Adventurer Card -----
7844
7845 Test 1 ---> Check if player draws 2 cards
7846
7847 Expected handCount = 6
7848 Actual handCount = 7
7849
7850 Test: FAILED
7851
7852
7853 Test 2 ---> Check if victory piles change
7854
7855 Expected province count = 8
7856 Actual province count = 8
7857
7858 Test: PASSED
7859
7860 Expected duchy count = 8
7861 Actual duchy count = 8
7862
7863 Test: PASSED
7864
7865 Expected estate count = 8
7866 Actual estate count = 8
7867
7868 Test: PASSED
7869
7870
7871 Test 3 ---> Check if kingdom card piles change
7872
7873 Testing card #1 ...
7874 Test: PASSED
7875
7876 Testing card #2 ...
7877 Test: PASSED
7878
7879 Testing card #3 ...
7880 Test: PASSED
7881
7882 Testing card #4 ...
7883 Test: PASSED
7884
7885 Testing card #5 ...
7886 Test: PASSED
7887
7888 Testing card #6 ...
7889 Test: PASSED
7890
7891 Testing card #7 ...
7892 Test: PASSED
7893
7894 Testing card #8 ...
7895 Test: PASSED
7896
7897 Testing card #9 ...
7898 Test: PASSED
7899
7900 Testing card #10 ...
7901 Test: PASSED
7902
```

D:\Users\Brian\Desktop\OSU\CS 362\Week 8\Assignment-5\Code\unittestresults.ou... — □ ×

File Edit Selection Find View Goto Tools Project Preferences Help

Makefile × cardtest1.c × cardtest2.c × cardtest3.c × cardtest4.c × ▾

```
7901 Test: PASSED
7902
7903
7904 Test 4 ---> Check if player gained 2 treasure cards
7905
7906 Test: PASSED
7907
7908
7909 Test 5 ---> Check if other player was affected
7910
7911 Expected deck count = 10
7912 Actual deck count = 10
7913 Test: PASSED
7914
7915 Expected hand count = 0
7916 Actual hand count = 0
7917 Test: PASSED
7918
7919
7920 TEST FAILED
7921 FAILS: 1
7922
7923 File 'cardtest1.c'
7924 Lines executed:98.46% of 65
7925 Branches executed:100.00% of 22
7926 Taken at least once:77.27% of 22
7927 Calls executed:97.50% of 40
7928 Creating 'cardtest1.c.gcov'
7929
7930 File 'dominion.c'
7931 Lines executed:19.89% of 558
7932 Branches executed:23.26% of 417
7933 Taken at least once:14.87% of 417
7934 Calls executed:10.53% of 95
7935 Creating 'dominion.c.gcov'
7936
```

.* Aa “ ” ☰ ☐ card ▾ Find Find Prev Find All ×

☐ 1 of 3 matches Spaces: 4 Plain Text

D:\Users\Brian\Desktop\OSU\CS 362\Week 8\Assignment-5\Code\unittestresults.out - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

Makefile x cardtest1.c x cardtest2.c x cardtest3.c x cardtest4.c x dominion.c x stadventurer.out x

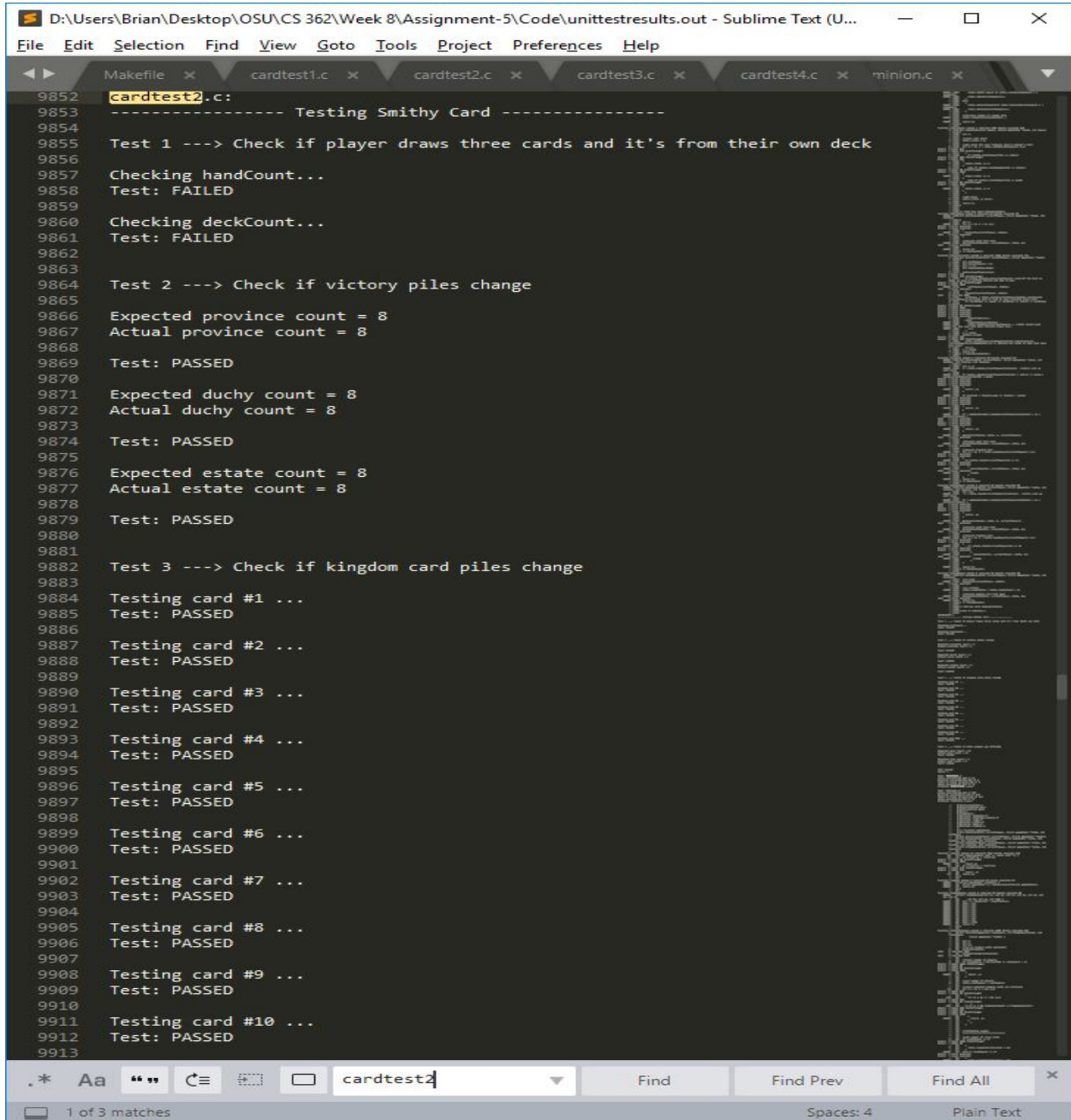
```
9693      -: 1263:
9694 function AdventurerCard called 1 returned 100% blocks executed 73%
9695 ..... 1: 1264: int AdventurerCard(int currentPlayer, struct gameState *state)
9696 ..... -: 1265: {
9697 ..... -: 1266:     int cardDrawn;
9698 ..... 1: 1267:     int drawntreasure = 0;
9699 ..... 1: 1268:     int z = 1;
9700 ..... -: 1269:     int temphand[MAX_HAND];
9701 ..... -: 1270:
9702 ..... 4: 1271:     while(drawntreasure<2){
9703 branch 0 taken 67%
9704 branch 1 taken 33% (fallthrough)
9705 ..... 2: 1272:     if (state->deckCount[currentPlayer] < 1){ //if the deck is empty we need to
          shuffle discard and add to deck
9706 branch 0 taken 0% (fallthrough)
9707 branch 1 taken 100%
9708 ..... #####: 1273:     shuffle(currentPlayer, state);
9709 call ... 0 never executed
9710 ..... -: 1274:     }
9711 ..... 2: 1275:     drawCard(currentPlayer, state);
9712 call ... 0 returned 100%
9713 ..... 2: 1276:     cardDrawn = state->hand[currentPlayer][state->handCount[currentPlayer]-1]; //
          top card of hand is most recently drawn card.
9714 ..... 2: 1277:     if (cardDrawn == copper || cardDrawn == silver || cardDrawn == gold)
9715 branch 0 taken 0% (fallthrough)
9716 branch 1 taken 100%
9717 branch 2 never executed
9718 branch 3 never executed
9719 branch 4 never executed
9720 branch 5 never executed
9721 ..... 2: 1278:     drawntreasure++;
9722 ..... -: 1279:     else{
9723 ..... #####: 1280:     temphand[z]=cardDrawn;
9724 ..... #####: 1281:     state->handCount[currentPlayer]--; //this should just remove the top card (
          the most recently drawn one).
9725 ..... #####: 1282:     z++;
9726 ..... -: 1283:     }
9727 ..... -: 1284:     } // while
9728 ..... 3: 1285:     while(z-1>0){
9729 branch 0 taken 50%
9730 branch 1 taken 50% (fallthrough)
9731 ..... 1: 1286:     state->discard[currentPlayer][state->discardCount[currentPlayer]++] = temphand[
          z-1]; // discard all cards in play that have been drawn
9732 ..... 1: 1287:     z=z-1;
9733 ..... -: 1288:     } // while
9734 ..... 1: 1289:     return 0;
9735 ..... -: 1290:     } // adventurerCard();
9736 ..... -: 1291: }
9737 function MineCard called 0 returned 0% blocks executed 0%
```

.* Aa " " ☰ ☐ cardtest Find Find Prev Find All x

☐ 43 lines, 1857 characters selected Spaces: 4 Plain Text

cardtest2.c

My cardtest2.c file was created to test the smithy card. My test received 100% statement and 100% branch coverage. The overall dominion.c coverage was only 20.43% statement and 23.26% branch coverage. One of the tests in cardtest2.c failed. The hand count and deck count were different from what they should have been.



```
9852 cardtest2.c:
9853 ----- Testing Smithy Card -----
9854
9855 Test 1 ---> Check if player draws three cards and it's from their own deck
9856
9857 Checking handCount...
9858 Test: FAILED
9859
9860 Checking deckCount...
9861 Test: FAILED
9862
9863
9864 Test 2 ---> Check if victory piles change
9865
9866 Expected province count = 8
9867 Actual province count = 8
9868
9869 Test: PASSED
9870
9871 Expected duchy count = 8
9872 Actual duchy count = 8
9873
9874 Test: PASSED
9875
9876 Expected estate count = 8
9877 Actual estate count = 8
9878
9879 Test: PASSED
9880
9881
9882 Test 3 ---> Check if kingdom card piles change
9883
9884 Testing card #1 ...
9885 Test: PASSED
9886
9887 Testing card #2 ...
9888 Test: PASSED
9889
9890 Testing card #3 ...
9891 Test: PASSED
9892
9893 Testing card #4 ...
9894 Test: PASSED
9895
9896 Testing card #5 ...
9897 Test: PASSED
9898
9899 Testing card #6 ...
9900 Test: PASSED
9901
9902 Testing card #7 ...
9903 Test: PASSED
9904
9905 Testing card #8 ...
9906 Test: PASSED
9907
9908 Testing card #9 ...
9909 Test: PASSED
9910
9911 Testing card #10 ...
9912 Test: PASSED
9913
```

D:\Users\Brian\Desktop\OSU\CS 362\Week 8\Assignment-5\Code\unittestresults.out - Sublime Text (U... — □ ×

File Edit Selection Find View Goto Tools Project Preferences Help

Makefile x cardtest1.c x cardtest2.c x cardtest3.c x cardtest4.c x minion.c x

```
9912 Test: PASSED
9913
9914
9915 Test 4 ---> Check if other player was affected
9916
9917 Expected deck count = 10
9918 Actual deck count = 10
9919 Test: PASSED
9920
9921 Expected hand count = 0
9922 Actual hand count = 0
9923 Test: PASSED
9924
9925
9926 TEST FAILED
9927 FAILS: 2
9928
9929 File 'cardtest2.c'
9930 Lines executed:98.15% of 54
9931 Branches executed:100.00% of 6
9932 Taken at least once:83.33% of 6
9933 Calls executed:97.44% of 39
9934 Creating 'cardtest2.c.gcov'
9935
9936 File 'dominion.c'
9937 Lines executed:20.43% of 558
9938 Branches executed:23.26% of 417
9939 Taken at least once:14.63% of 417
9940 Calls executed:11.58% of 95
9941 Creating 'dominion.c.gcov'
9942
```

.* Aa "" ☰ ☐ cardtest2 Find Find Prev Find All x

1 of 3 matches Spaces: 4 Plain Text

D:\Users\Brian\Desktop\OSU\CS 362\Week 8\Assignment-5\Code\unittestresults.out - Sublime Text (UNREGISTERED) — □ ×

File Edit Selection Find View Goto Tools Project Preferences Help

Makefile x cardtest1.c x cardtest2.c x cardtest3.c x cardtest4.c x dominion.c x irer.c x

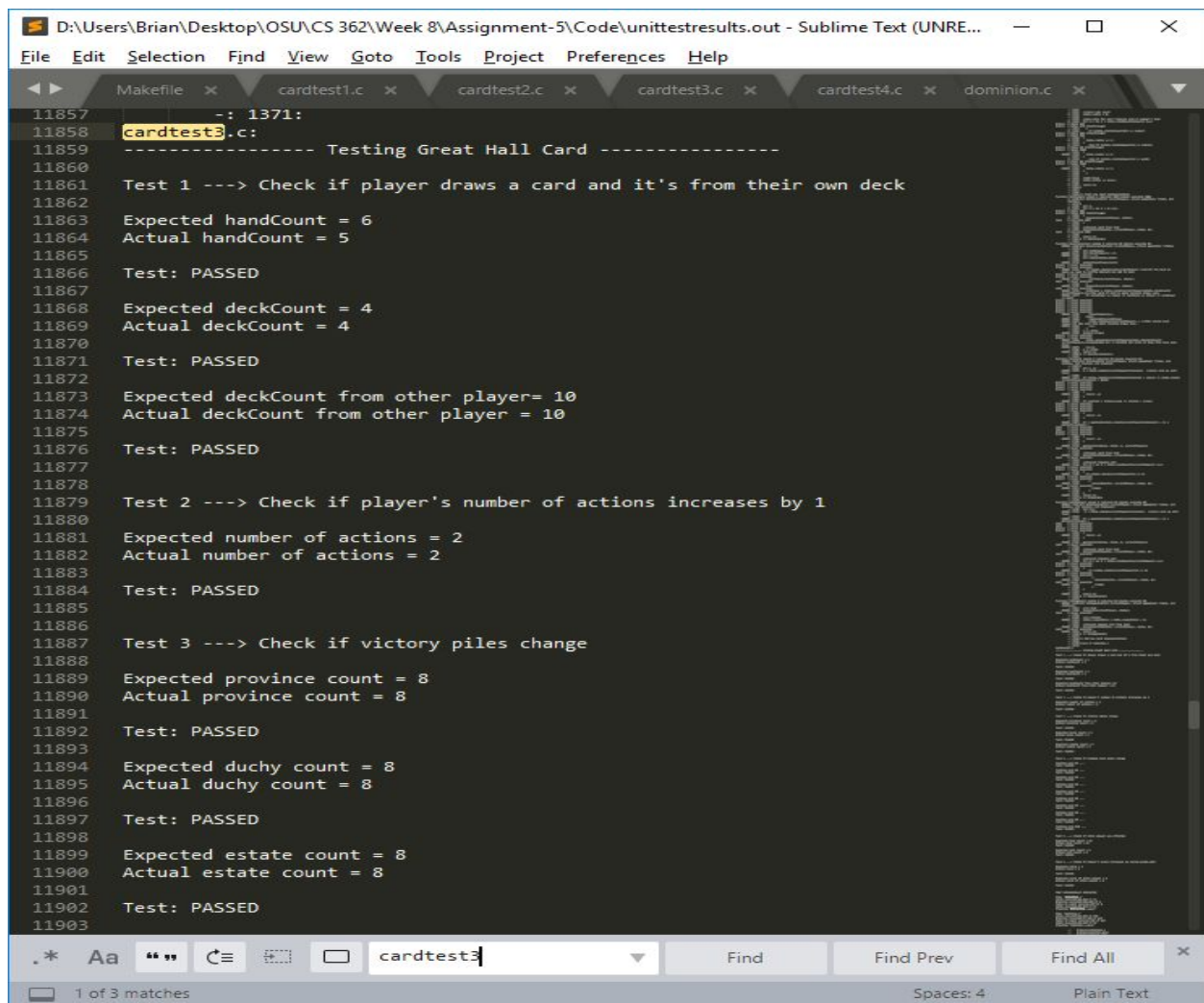
```
11680 --: 1249:// START New Card Implementations
11681 function SmithyCard called 1 returned 100% blocks executed 100%
11682 .....1: 1250:int SmithyCard(int currentPlayer, struct gameState *state, int handPos)
11683 .....--: 1251:{
11684 .....--: 1252: int i;
11685 .....5: 1253: for (i = 0; i < 4; i++)
11686 branch 0 taken 80%
11687 branch 1 taken 20% (fallthrough)
11688 .....--: 1254:{
11689 .....4: 1255: drawCard(currentPlayer, state);
11690 call 0 returned 100%
11691 .....--: 1256:}
11692 .....--: 1257:
11693 .....--: 1258://discard card from hand
11694 .....1: 1259: discardCard(handPos, currentPlayer, state, 0);
11695 call 0 returned 100%
11696 .....--: 1260:
11697 .....1: 1261: return 0;
11698 .....--: 1262:} // SmithyCard()
11699 .....--: 1263:
11700 function AdventurerCard called 0 returned 0% blocks executed 0%
11701 #####: 1264:int AdventurerCard(int currentPlayer, struct gameState *state)
```

.* Aa "" ☰ ☐ cardtest3 Find Find Prev Find All x

3 matches Spaces: 4 Plain Text

cardtest3.c

My cardtest3.c was created to test the great hall card. Unfortunately, my teammate did not refactor this card or make a separate function for this card so I was not given a definitive statement coverage. Although, it is a very simple card so after inspecting the gcov, it appears that the test received 100% statement and 100% branch coverage. The dominion.c coverage was 22.22% statement coverage and 27.10% branch coverage. There were no problem with this function as it was not refactored by my teammate and passed all the tests.



```
11857 -: 1371:
11858 cardtest3.c:
11859 ----- Testing Great Hall Card -----
11860
11861 Test 1 ---> Check if player draws a card and it's from their own deck
11862
11863 Expected handCount = 6
11864 Actual handCount = 5
11865
11866 Test: PASSED
11867
11868 Expected deckCount = 4
11869 Actual deckCount = 4
11870
11871 Test: PASSED
11872
11873 Expected deckCount from other player= 10
11874 Actual deckCount from other player = 10
11875
11876 Test: PASSED
11877
11878
11879 Test 2 ---> Check if player's number of actions increases by 1
11880
11881 Expected number of actions = 2
11882 Actual number of actions = 2
11883
11884 Test: PASSED
11885
11886
11887 Test 3 ---> Check if victory piles change
11888
11889 Expected province count = 8
11890 Actual province count = 8
11891
11892 Test: PASSED
11893
11894 Expected duchy count = 8
11895 Actual duchy count = 8
11896
11897 Test: PASSED
11898
11899 Expected estate count = 8
11900 Actual estate count = 8
11901
11902 Test: PASSED
11903
```


D:\Users\Brian\Desktop\OSU\CS 362\Week 8\Assignment-5\Code\unittestresults.out - Sublime Text (UNRE... — □ ×

File Edit Selection Find View Goto Tools Project Preferences Help

Makefile x cardtest1.c x cardtest2.c x cardtest3.c x cardtest4.c x dominion.c x

```
11905 Test 4 ----> Check if kingdom card piles change
11906
11907 Testing card #1 ...
11908 Test: PASSED
11909
11910 Testing card #2 ...
11911 Test: PASSED
11912
11913 Testing card #3 ...
11914 Test: PASSED
11915
11916 Testing card #4 ...
11917 Test: PASSED
11918
11919 Testing card #5 ...
11920 Test: PASSED
11921
11922 Testing card #6 ...
11923 Test: PASSED
11924
11925 Testing card #7 ...
11926 Test: PASSED
11927
11928 Testing card #8 ...
11929 Test: PASSED
11930
11931 Testing card #9 ...
11932 Test: PASSED
11933
11934 Testing card #10 ...
11935 Test: PASSED
11936
11937
11938 Test 5 ----> Check if other player was affected
11939
11940 Expected deck count = 10
11941 Actual deck count = 10
11942 Test: PASSED
11943
11944 Expected hand count = 0
11945 Actual hand count = 0
11946 Test: PASSED
11947
11948
11949 Test 6 ----> Check if player's score increases by having great_hall
11950
11951 Expected score = 2
11952 Actual score = 1
11953
11954 Test: PASSED
11955
11956 Expected score of other player = 0
11957 Actual score of other player = 0
11958
11959 Test: PASSED
11960
11961
11962 TEST SUCCESSFULLY COMPLETED
11963
11964 File 'cardtest3.c'
11965 Lines executed:94.59% of 74
11966 Branches executed:100.00% of 6
11967 Taken at least once:66.67% of 6
11968 Calls executed:95.52% of 67
11969 Creating 'cardtest3.c.gcov'
11970
11971 File 'dominion.c'
11972 Lines executed:22.22% of 558
11973 Branches executed:27.10% of 417
11974 Taken at least once:16.79% of 417
11975 Calls executed:10.53% of 95
11976 Creating 'dominion.c.gcov'
11977
```

.* Aa "" C≡ 1 of 3 matches cardtest3 Find Find Prev Find All Spaces: 4 Plain Text

D:\Users\Brian\Desktop\OSU\CS 362\Week 8\Assignment-5\Code\unittestresults.out - Sublime Text (UNRE... — □ ×

File Edit Selection Find View Goto Tools Project Preferences Help

Makefile x cardtest1.c x cardtest2.c x cardtest3.c x cardtest4.c x dominion.c x

```
13121 -: 810:
13122 #####: 819: return 0;
13123 -: 820:
13124 -: 821: case great_hall:
13125 ..... -: 822: .....//+1 Card
13126 ..... 1: 823: .....drawCard(currentPlayer, state);
13127 call...0 returned 100%
13128 ..... -: 824: .....
13129 ..... -: 825: .....//+1 Actions
13130 ..... 1: 826: .....state->numActions++;
13131 ..... -: 827: .....
13132 ..... -: 828: .....//discard card from hand
13133 ..... 1: 829: .....discardCard(handPos, currentPlayer, state, 0);
13134 call...0 returned 100%
13135 ..... 1: 830: .....return 0;
13136 -: 831:
13137 -: 832: case minion:
```

.* Aa "" C≡ 3 matches cardtest4 Find Find Prev Find All Spaces: 4 Plain Text

cardtest4.c

My cardtest4.c was created to test the council room card. Unfortunately, my teammate did not refactor this card or make a separate function for this card so I was not given a definitive statement coverage. Although, after inspecting the gcov, it appears that the test received 100% statement and 100% branch coverage. The overall coverage of dominion.c was 20.79% statement coverage and 24.22% branch coverage. There were no problem with this function as it was not refactored by my teammate and passed all the tests.

```
D:\Users\Brian\Desktop\OSU\CS 362\Week 8\Assignment-5\Code\unittestresults.out - Sublime Text (UNRE...
File Edit Selection Find View Goto Tools Project Preferences Help
Makefile cardtest1.c cardtest2.c cardtest3.c cardtest4.c dominion.c
13891 -: 1370://end of dominion.c
13892 -: 1371:
13893 cardtest4.c:
13894 ----- Testing Council Room Card -----
13895
13896 Test 1 ---> Check if player draws 4 cards and it's from their own deck
13897
13898 Expected handCount = 9
13899 Actual handCount = 8
13900
13901 Test: PASSED
13902
13903 Expected deckCount = 1
13904 Actual deckCount = 1
13905
13906 Test: PASSED
13907
13908
13909 Test 2 ---> Check if other player draws a card and it's from their own deck
13910
13911 Expected handCount = 1
13912 Actual handCount = 1
13913
13914 Test: PASSED
13915
13916 Expected deckCount = 9
13917 Actual deckCount = 9
13918
13919 Test: PASSED
13920
13921
13922 Test 3 ---> Check if player's number of buys increases by 1
13923
13924 Expected number of buys = 2
13925 Actual number of buys = 2
13926
13927 Test: PASSED
13928
13929
```

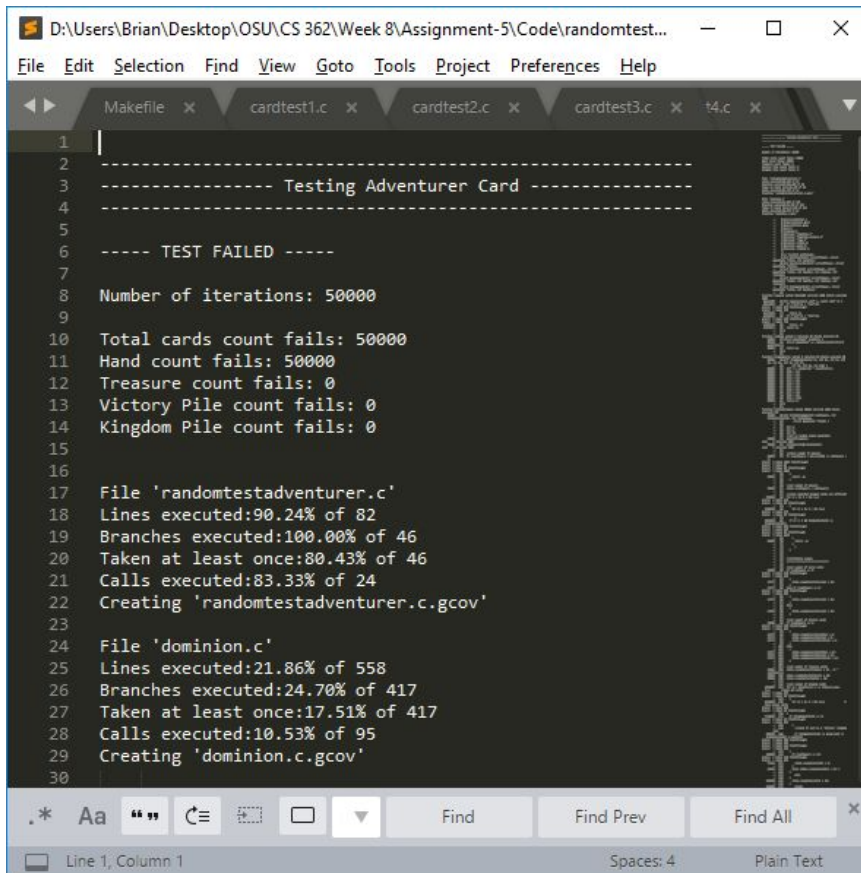
```
D:\Users\Brian\Desktop\OSU\CS 362\Week 8\Assignment-5\Code\un...
File Edit Selection Find View Goto Tools Project Preferences Help
Makefile x cardtest1.c x cardtest2.c x cardtest3.c x
13930 Test 4 ---> Check if victory piles change
13931
13932 Expected province count = 8
13933 Actual province count = 8
13934
13935 Test: PASSED
13936
13937 Expected duchy count = 8
13938 Actual duchy count = 8
13939
13940 Test: PASSED
13941
13942 Expected estate count = 8
13943 Actual estate count = 8
13944
13945 Test: PASSED
13946
13947
13948 Test 5 ---> Check if kingdom card piles change
13949
13950 Testing card #1 ...
13951 Test: PASSED
13952
13953 Testing card #2 ...
13954 Test: PASSED
13955
13956 Testing card #3 ...
13957 Test: PASSED
13958
13959 Testing card #4 ...
13960 Test: PASSED
13961
13962 Testing card #5 ...
13963 Test: PASSED
13964
13965 Testing card #6 ...
13966 Test: PASSED
13967
13968 Testing card #7 ...
13969 Test: PASSED
13970
13971 Testing card #8 ...
13972 Test: PASSED
13973
13974 Testing card #9 ...
13975 Test: PASSED
13976
13977 Testing card #10 ...
13978 Test: PASSED
13979
13980
13981 TEST SUCCESSFULLY COMPLETED
13982
13983 File 'cardtest4.c'
13984 Lines executed:93.44% of 61
13985 Branches executed:100.00% of 6
13986 Taken at least once:66.67% of 6
13987 Calls executed:93.48% of 46
13988 Creating 'cardtest4.c.gcov'
13989
13990 File 'dominion.c'
13991 Lines executed:20.79% of 558
13992 Branches executed:24.22% of 417
13993 Taken at least once:15.59% of 417
13994 Calls executed:11.58% of 95
13995 Creating 'dominion.c.gcov'
13996
```

.* Aa " " ☰ ☲ ☳ ☴ ☵ ☶ ☷ Find Find Prev Find All
1 of 3 matches Spaces: 4 Plain Text

D:\Users\Brian\Desktop\OSU\CS 362\Week 8\Assignment-5\Code\unittestresults.out - Sublime Tex...
File Edit Selection Find View Goto Tools Project Preferences Help
Makefile x cardtest1.c x cardtest2.c x cardtest3.c x cardtest4.c x n.c x
14943 call 0 never executed
14944 -: 676:
14945 -: 677: case council_room:
14946 -: 678: //+4 Cards
14947 5: 679: for (i = 0; i < 4; i++)
14948 branch 0 taken 80%
14949 branch 1 taken 20% (fallthrough)
14950 -: 680: {
14951 4: 681: drawCard(currentPlayer, state);
14952 call 0 returned 100%
14953 -: 682: }
14954 -: 683:
14955 -: 684: //+1 Buy
14956 1: 685: state->numBuys++;
14957 -: 686:
14958 -: 687: //Each other player draws a card
14959 3: 688: for (i = 0; i < state->numPlayers; i++)
14960 branch 0 taken 67%
14961 branch 1 taken 33% (fallthrough)
14962 -: 689: {
14963 2: 690: if (i != currentPlayer)
14964 branch 0 taken 50% (fallthrough)
14965 branch 1 taken 50%
14966 -: 691: {
14967 1: 692: drawCard(i, state);
14968 call 0 returned 100%
14969 -: 693: }
14970 -: 694: }
14971 -: 695:
14972 -: 696: //put played card in played card pile
14973 1: 697: discardCard(handPos, currentPlayer, state, 0);
14974 call 0 returned 100%
14975 -: 698:
14976 1: 699: return 0;
14977 -: 700:
14978 -: 701: case feast:
.* Aa " " ☰ ☲ ☳ ☴ ☵ ☶ ☷ cardtest4 Find Find Prev Find All x
3 matches Spaces: 4 Plain Text

randomtestadventurer.c

For the adventurer card, I completed 93% statement coverage and completely missed one if statement in the branch coverage. The coverage for the whole dominion.c file was 21.86% statement coverage and 24.70% branch coverage. The reason that I didn't achieve 100% statement and branch coverage for the adventurer card because I failed to hit the if statement which checks to see if the player's deck count is below one and then proceeds to shuffle the cards. I didn't hit this branch because I didn't allow the randomized deck count for the players go below 1 card because I wanted the deck to contain a certain amount of treasure cards to be able to test the card properly. This may have been an oversight and if I were to recreate this test I would allow the randomized deck count to hit zero instead of giving it a minimum amount. The same failures occurred for 50,000 iterations. The failure was that the total cards count was wrong and the hand count was wrong.



```
1 |
2 |
3 | ----- Testing Adventurer Card -----
4 | -----
5 |
6 | ----- TEST FAILED -----
7 |
8 | Number of iterations: 50000
9 |
10 | Total cards count fails: 50000
11 | Hand count fails: 50000
12 | Treasure count fails: 0
13 | Victory Pile count fails: 0
14 | Kingdom Pile count fails: 0
15 |
16 |
17 | File 'randomtestadventurer.c'
18 | Lines executed:90.24% of 82
19 | Branches executed:100.00% of 46
20 | Taken at least once:80.43% of 46
21 | Calls executed:83.33% of 24
22 | Creating 'randomtestadventurer.c.gcov'
23 |
24 | File 'dominion.c'
25 | Lines executed:21.86% of 558
26 | Branches executed:24.70% of 417
27 | Taken at least once:17.51% of 417
28 | Calls executed:10.53% of 95
29 | Creating 'dominion.c.gcov'
30 |
```


D:\Users\Brian\Desktop\OSU\CS 362\Week 8\Assignment-5\Code\randomtestadventurer.out - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

Makefile x cardtest1.c x cardtest2.c x cardtest3.c x cardtest4.c x dominion.c x randomtestadventurer.out x

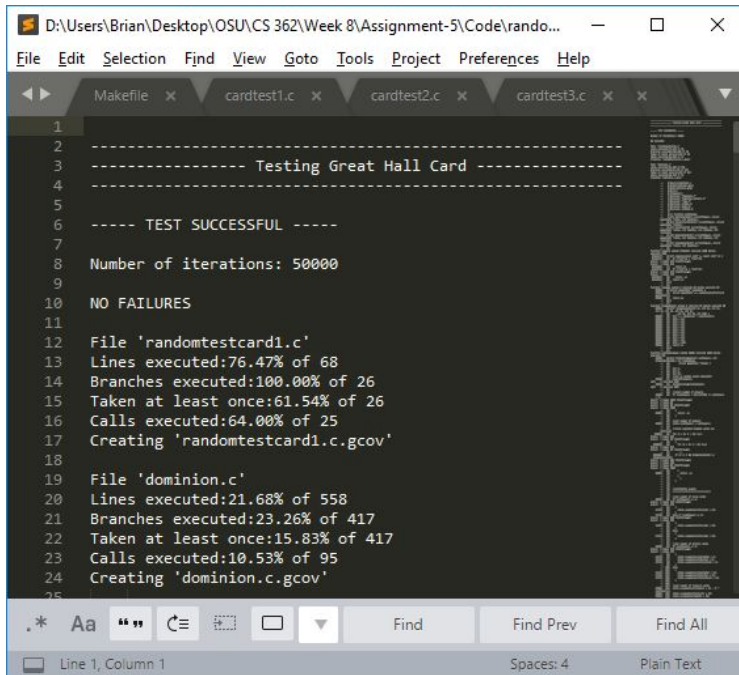
```
1786 // 1262: // emptyCard()
1787 -: 1263:
1788 function AdventurerCard called 50000 returned 100% blocks executed 93%
1789 ... 50000: 1264: int AdventurerCard(int currentPlayer, struct gameState *state)
1790 ..... -: 1265: {
1791 ..... -: 1266:     int cardDrawn;
1792 ... 50000: 1267:     int drawntreasure = 0;
1793 ... 50000: 1268:     int z = 1;
1794 ..... -: 1269:     int temphand[MAX_HAND];
1795 ..... -: 1270:
1796 ... 6671321: 1271:     while(drawntreasure<2){
1797 branch· 0 taken 99%
1798 branch· 1 taken 1% (fallthrough)
1799 ... 6571321: 1272:     if (state->deckCount[currentPlayer]<1){ //if the deck is empty we need to shuffle
1800     discard and add to deck
1801 branch· 0 taken 0% (fallthrough)
1802 branch· 1 taken 100%
1803 ... #####: 1273:     shuffle(currentPlayer, state);
1804 call···· 0 never executed
1805 ..... -: 1274:     }
1806 ... 6571321: 1275:     drawCard(currentPlayer, state);
1807 call···· 0 returned 100%
1808 ... 6571321: 1276:     cardDrawn = state->hand[currentPlayer][state->handCount[currentPlayer]-1]; //top card of
1809     hand is most recently drawn card.
1810 ... 6571321: 1277:     if (cardDrawn == copper || cardDrawn == silver || cardDrawn == gold)
1811 branch· 0 taken 99% (fallthrough)
1812 branch· 1 taken 1%
1813 branch· 2 taken 99% (fallthrough)
1814 branch· 3 taken 1%
1815 branch· 4 taken 1% (fallthrough)
1816 branch· 5 taken 99%
1817 ... 100000: 1278:     drawntreasure++;
1818 ..... -: 1279:     else{
1819 ... 6471321: 1280:     temphand[z]=cardDrawn;
1820 ... 6471321: 1281:     state->handCount[currentPlayer]--; //this should just remove the top card (the most
1821     recently drawn one).
1822 ... 6471321: 1282:     z++;
1823 ..... -: 1283:     }
1824 ..... -: 1284:     } // while
1825 ... 6621321: 1285:     while(z-1>=0){
1826 branch· 0 taken 99%
1827 branch· 1 taken 1% (fallthrough)
1828 ... 6521321: 1286:     state->discard[currentPlayer][state->discardCount[currentPlayer]++]=temphand[z-1]; //
1829     discard all cards in play that have been drawn
1830 ... 6521321: 1287:     z=z-1;
1831 ..... -: 1288:     } // while
1832 ... 50000: 1289:     return 0;
1833 ... 50000: 1290: } // adventurerCard();
1834 ..... -: 1291:
1835 function MineCard called 0 returned 0% blocks executed 0%
1836 #####: 1292: int MineCard(int currentPlayer, struct gameState *state, int handPos, int choice1, int
```

.* Aa " ¶ cardtest4 Find Find Prev Find All x

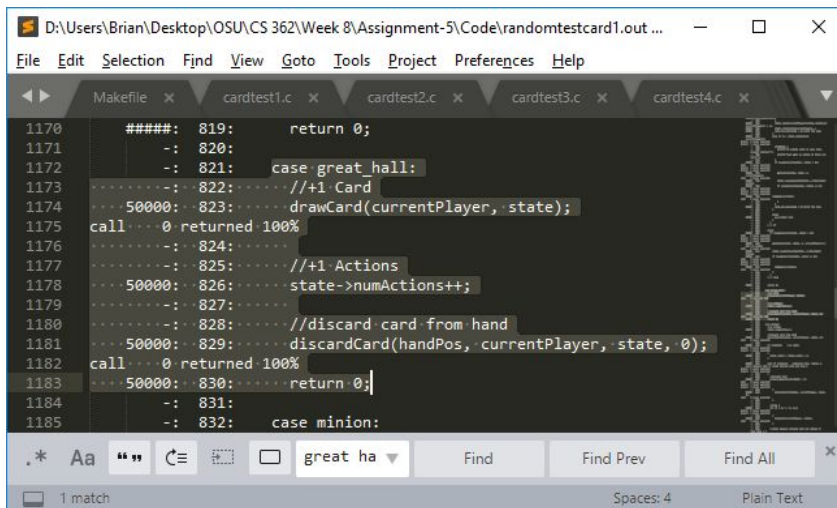
43 lines, 1864 characters selected Spaces: 4 Plain Text

randomtestcard1.c

For randomtestcard1.c, I tested the Great Hall card. The test received 100% statement and branch coverage for the Great Hall card function. The Great Hall card is a very basic card function that only has one branch and my teammate didn't refactor this card with a bug or it's own function, so I am not surprised that I achieved 100% statement and branch coverage. It only took 1 iteration to complete 100% coverage. It would have been very hard to not hit 100% coverage for this function. All the tests that I created passed for this function.



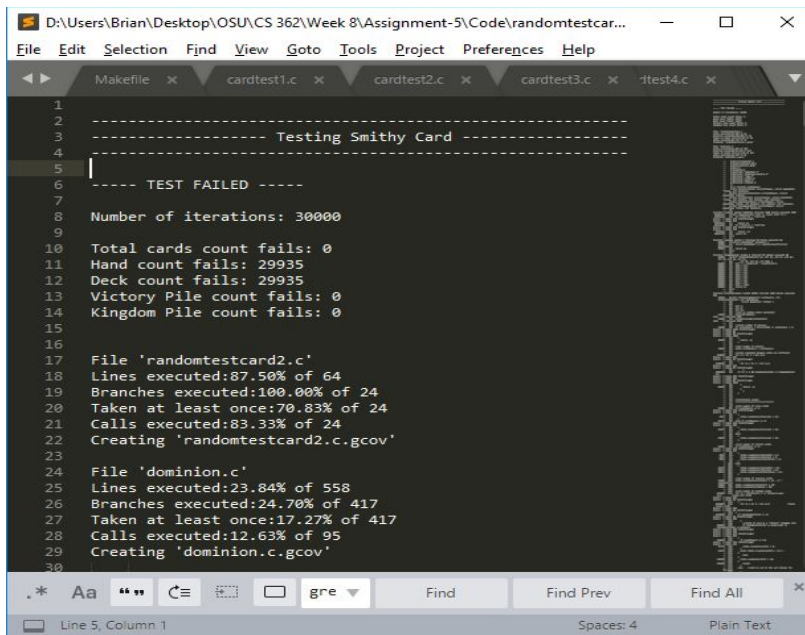
```
1
2 -----
3 ----- Testing Great Hall Card -----
4 -----
5
6 ----- TEST SUCCESSFUL -----
7
8 Number of iterations: 50000
9
10 NO FAILURES
11
12 File 'randomtestcard1.c'
13 Lines executed:76.47% of 68
14 Branches executed:100.00% of 26
15 Taken at least once:61.54% of 26
16 Calls executed:64.00% of 25
17 Creating 'randomtestcard1.c.gcov'
18
19 File 'dominion.c'
20 Lines executed:21.68% of 558
21 Branches executed:23.26% of 417
22 Taken at least once:15.83% of 417
23 Calls executed:10.53% of 95
24 Creating 'dominion.c.gcov'
25
```



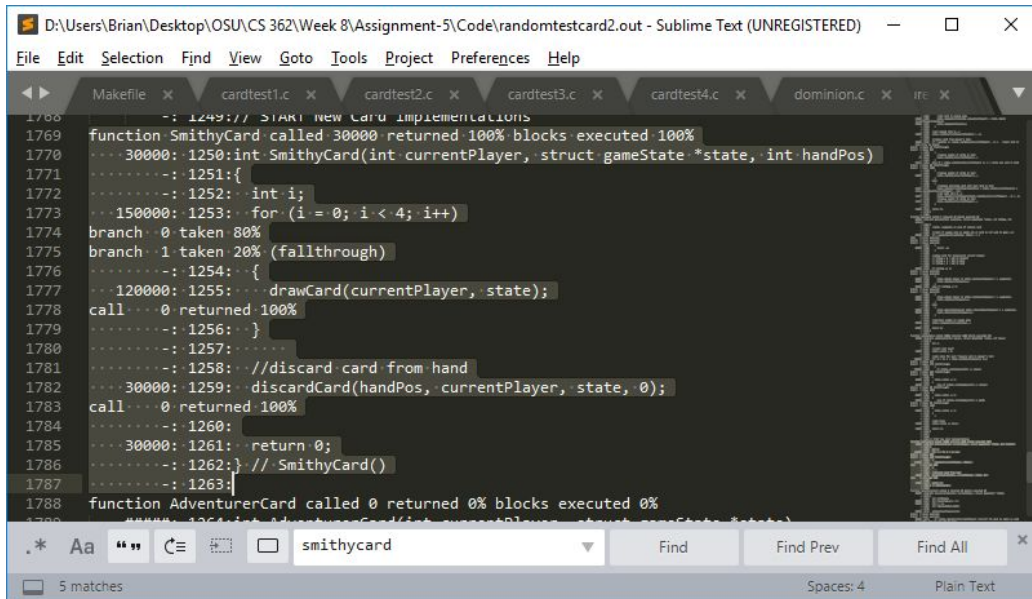
```
1170 #####: 819: return 0;
1171 -: 820:
1172 -: 821: case great_hall:
1173 .....: 822: .....//+1 Card
1174 ...50000: 823: .....drawCard(currentPlayer, state);
1175 call...0 returned 100%
1176 .....: 824: .....
1177 .....: 825: .....//+1 Actions
1178 ...50000: 826: .....state->numActions++;
1179 .....: 827: .....
1180 .....: 828: .....discard card from hand
1181 ...50000: 829: .....discardCard(handPos, currentPlayer, state, 0);
1182 call...0 returned 100%
1183 ...50000: 830: .....return 0;
1184 -: 831:
1185 -: 832: case minion:
```

randomtestcard2.c

For randomtestcard2.c, I tested the Smithy card. The test received 100% statement and 100% branch coverage. The overall dominion.c coverage was 23.84% statement and 34.70% branch coverage. 99% of the time the test failed in hand and deck count.



```
1
2 -----
3 ----- Testing Smithy Card -----
4 -----
5
6 ----- TEST FAILED -----
7
8 Number of iterations: 30000
9
10 Total cards count fails: 0
11 Hand count fails: 29935
12 Deck count fails: 29935
13 Victory Pile count fails: 0
14 Kingdom Pile count fails: 0
15
16
17 File 'randomtestcard2.c'
18 Lines executed:87.50% of 64
19 Branches executed:100.00% of 24
20 Taken at least once:70.83% of 24
21 Calls executed:83.33% of 24
22 Creating 'randomtestcard2.c.gcov'
23
24 File 'dominion.c'
25 Lines executed:23.84% of 558
26 Branches executed:24.70% of 417
27 Taken at least once:17.27% of 417
28 Calls executed:12.63% of 95
29 Creating 'dominion.c.gcov'
30
```

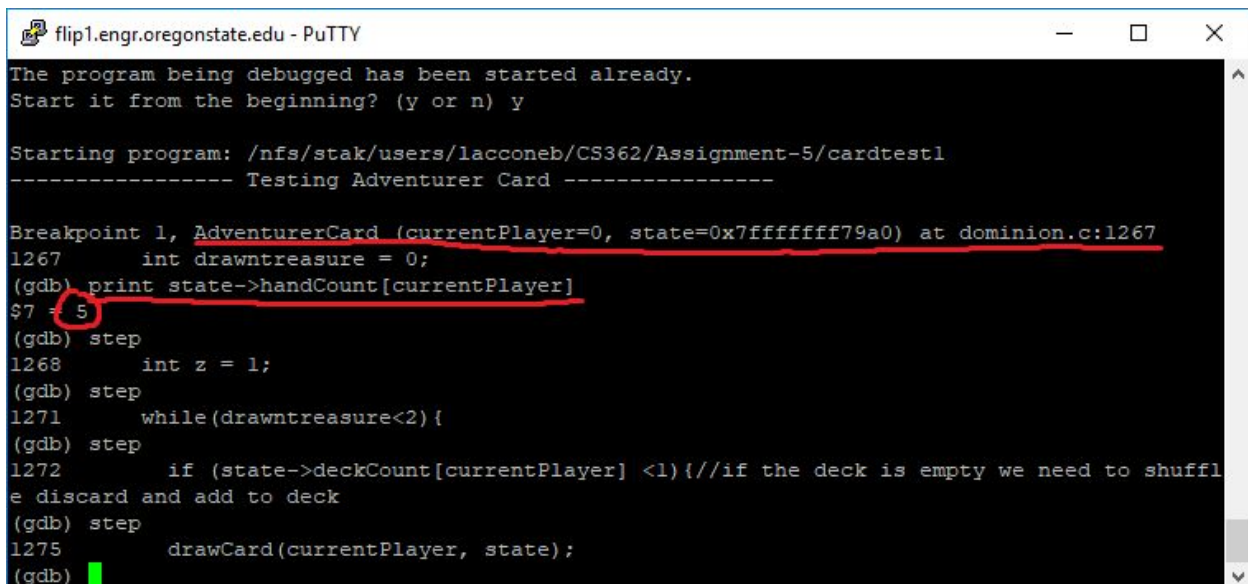


```
1769 function SmithyCard called 30000 returned 100% blocks executed 100%
1770 ... 30000: 1250: int SmithyCard(int currentPlayer, struct gameState *state, int handPos)
1771 ... 1251: {
1772 ... 1252:     int i;
1773 ... 150000: 1253:     for (i = 0; i < 4; i++)
1774 branch 0 taken 80%
1775 branch 1 taken 20% (fallthrough)
1776 ... 1254: {
1777 ... 120000: 1255:     drawCard(currentPlayer, state);
1778 call ... 0 returned 100%
1779 ... 1256: }
1780 ... 1257: }
1781 ... 1258: //discard card from hand
1782 ... 30000: 1259:     discardCard(handPos, currentPlayer, state, 0);
1783 call ... 0 returned 100%
1784 ... 1260: }
1785 ... 30000: 1261:     return 0;
1786 ... 1262: } // SmithyCard()
1787 ... 1263: }
1788 function AdventurerCard called 0 returned 0% blocks executed 0%
```

Debugging

In this section I will be going over how I debugged the three bugs that I found while testing my teammate's code.

1. The first bug that I noticed was the hand count difference while testing cardtest1.c (AdventurerCard()). I noticed that the hand count was always 1 higher so I used gdb to debug the problem. In the following pictures I have displayed how I came to realize how the bug was happening. I didn't capture the whole process because it was many lines of stepping through the function. I first made a breakpoint at AdventurerCard in cardtest1.c. I ran gdb on cardtest1 and once it hit the breakpoint I printed out the hand count of the current player. It printed 5, which is what is expected because players start the game with 5 cards in hand. Next, I stepped through the whole function and printed the hand count at certain times throughout the function. It drew 2 treasure cards and hit 7 cards. That hand count stayed the same until the end of the function. After debugging this problem, I quickly realized that the adventurer card, which is in hand, is never being discarded. To fix this bug, I added another parameter to the function called handPos to hold the hand position of the adventurer card and then sent a call to discardCard() to discard the adventurer card. This fixed the bug and the preceding test verified that the hand counts were the same.



```
flip1.engr.oregonstate.edu - PuTTY
The program being debugged has been started already.
Start it from the beginning? (y or n) y

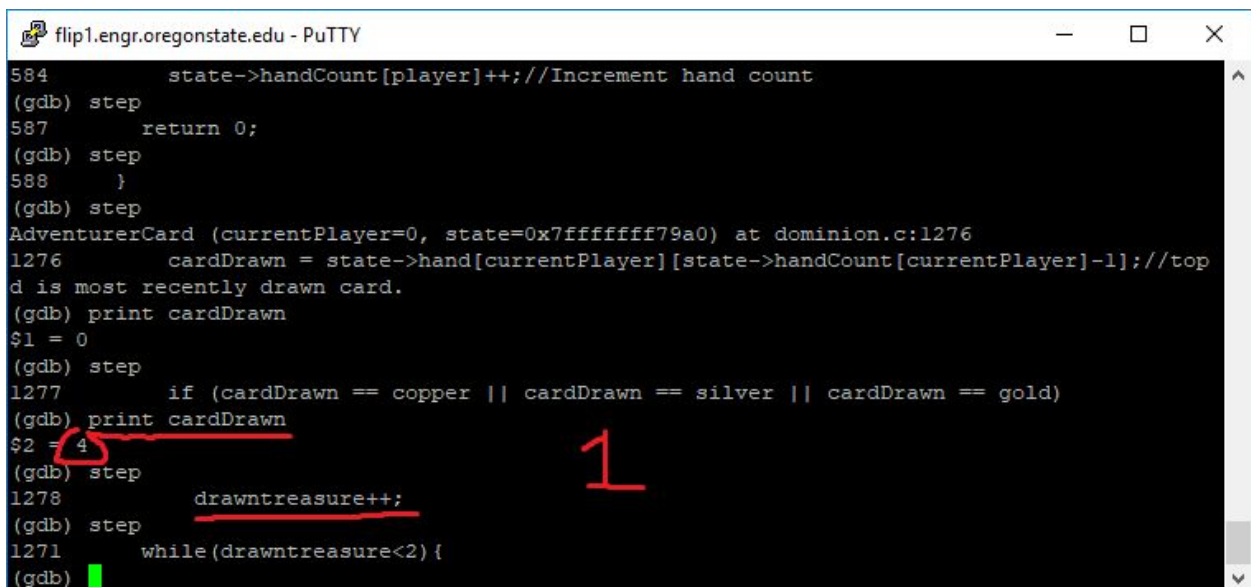
Starting program: /nfs/stak/users/lacconeb/CS362/Assignment-5/cardtest1
----- Testing Adventurer Card -----

Breakpoint 1, AdventurerCard (currentPlayer=0, state=0x7fffffff79a0) at dominion.c:1267
1267     int drawntreasure = 0;
(gdb) print state->handCount[currentPlayer]
$7 = 5
(gdb) step
1268     int z = 1;
(gdb) step
1271     while(drawntreasure<2){
(gdb) step
1272         if (state->deckCount[currentPlayer] <1){//if the deck is empty we need to shuffle
e discard and add to deck
(gdb) step
1275         drawCard(currentPlayer, state);
(gdb)
```



```
flip1.engr.oregonstate.edu - PuTTY
(gdb) step
1285     while (z-1>=0){
(gdb) print z
$4 = 1
(gdb) step
1286     state->discard[currentPlayer][state->discardCount[currentPlayer]++]=temphand[z-1
]; // discard all cards in play that have been drawn
(gdb) print state->handCount[currentPlayer]
$5 = 7
(gdb) step
1287     z=z-1;
(gdb) step
1285     while (z-1>=0){
(gdb) step
1289     return 0;
(gdb) step
1290     } // adventurerCard();
(gdb) print state->handCount[currentPlayer]
$6 = 7
(gdb)
```

2. The second bug that I found while testing `AdventurerCard()` was the total card count problem that `randomtestadventurer.c` discovered. This test was concluding that the sum of the number of cards in hand, discard pile, and deck were not the same after calling `AdventurerCard()` than before the function ran. While running through the function with `gdb`, I made sure to keep track of what branch the code was taking. The first and second card drawn was a treasure card, which means that the function never took the path to lower the hand count and increment `int z` by one. This is highlighted in the first two pictures in this section. The third picture shows the value of `z` after the `while(drawntreasure<2)` loop finished. The value of `z` was 1, when the value should have been 0 to prevent the function from discarding the non treasure cards that the function drew. I quickly looked up at the value of `z` and realized that my teammate had initialized `int z` to equal 1 instead of 0. To fix this problem, I simply initialized `int z = 0`. After fixing this mistake, both `cardtest1.c` and `randomtestadventurer.c` were successful.



```
flip1.engr.oregonstate.edu - PuTTY
584      state->handCount[player]++; //Increment hand count
(gdb) step
587      return 0;
(gdb) step
588  }
(gdb) step
AdventurerCard (currentPlayer=0, state=0x7fffffff79a0) at dominion.c:1276
1276      cardDrawn = state->hand[currentPlayer][state->handCount[currentPlayer]-1]; //top
d is most recently drawn card.
(gdb) print cardDrawn
$1 = 0
(gdb) step
1277      if (cardDrawn == copper || cardDrawn == silver || cardDrawn == gold)
(gdb) print cardDrawn
$2 = 4
(gdb) step
1278      drawntreasure++;
(gdb) step
1271      while (drawntreasure<2) {
(gdb)
```

```
flip1.engr.oregonstate.edu - PuTTY
583     state->deckCount[player]--;
(gdb) step
584     state->handCount[player]++; //Increment hand count
(gdb) step
587     return 0;
(gdb) step
588 }
(gdb) step
AdventurerCard (currentPlayer=0, state=0x7fffffff79a0) at dominion.c:1276
1276     cardDrawn = state->hand[currentPlayer][state->handCount[currentPlayer]-1]; //top
card of hand is most recently drawn card.
(gdb) step
1277     if (cardDrawn == copper || cardDrawn == silver || cardDrawn == gold)
(gdb) print cardDrawn
$3 = 4
(gdb) step
1278     drawntreasure++;
(gdb) step
1271     while(drawntreasure<2){
(gdb)
```

```
flip1.engr.oregonstate.edu - PuTTY
(gdb) step
AdventurerCard (currentPlayer=0, state=0x7fffffff79a0) at dominion.c:1276
1276     cardDrawn = state->hand[currentPlayer][state->handCount[currentPlayer]-1]; //top
card of hand is most recently drawn card.
(gdb) step
1277     if (cardDrawn == copper || cardDrawn == silver || cardDrawn == gold)
(gdb) print cardDrawn
$3 = 4
(gdb) step
1278     drawntreasure++;
(gdb) step
1271     while(drawntreasure<2){
(gdb) step
1285     while(z-1>=0){
(gdb) print z
$4 = 1
(gdb) step
1286     state->discard[currentPlayer][state->discardCount[currentPlayer]++]=temphand[z-1];
// discard all cards in play that have been drawn
(gdb)
```

3. The third and last bug that I discovered was found when running `cardeffect2.c` and `randomtestcard2.c`. This test discovered that there was a hand count error after running `SmithyCard()`. The hand count was always one more than it should be except for the rare random test which only initialized the starting deck to have 3 cards in it. The following screenshots illustrate the hand count before and after running `SmithyCard()`. The hand count was 8 instead of 7. During the for loop I also printed the hand count after each iteration ran and realized that the loop was running 4 times before it stopped. This confirmed that the for loop which was suppose to only draw three cards was drawing 4 instead. To fix this problem I simply changed the for loop to `i<3` instead of `i<4`. This fixed the problem and all subsequent tests were successful.

```
flip1.engr.oregonstate.edu - PuTTY
----- Testing Smithy Card -----
Breakpoint 1, SmithyCard (currentPlayer=0, state=0x7fffffff79a0, handPos=0)
  at dominion.c:1253
1253     for (i = 0; i < 4; i++)
(gdb) print state->handCount[currentPlayer]
$1 = 5
(gdb) step
1255     drawCard(currentPlayer, state);
(gdb) step
drawCard (player=0, state=0x7fffffff79a0) at dominion.c:535
535     if (state->deckCount[player] <= 0) { //Deck is empty
(gdb) step
575     int count = state->handCount[player]; //Get current hand count for player
(gdb) step
581     deckCounter = state->deckCount[player]; //Create holder for the deck count
(gdb) step
582     state->hand[player][count] = state->deck[player][deckCounter - 1]; //Add card to
the hand
(gdb)
```

```
flip1.engr.oregonstate.edu - PuTTY
1163     else if ( state->handCount[currentPlayer] == 1 ) //only one card in hand
(gdb) step
1171     state->hand[currentPlayer][handPos] = state->hand[currentPlayer][ (state->hand
Count[currentPlayer] - 1)];
(gdb) step
1173     state->hand[currentPlayer][state->handCount[currentPlayer] - 1] = -1;
(gdb) step
1175     state->handCount[currentPlayer]--;
(gdb) step
1178     return 0;
(gdb) step
1179     }
(gdb) step
SmithyCard (currentPlayer=0, state=0x7fffffff79a0, handPos=0) at dominion.c:1261
1261     return 0;
(gdb) step
1262     } // SmithyCard()
(gdb) print state->handCount[currentPlayer]
$2 = 8
(gdb)
```