**Design of Classes**

For the Board class, I used dynamically allocated arrays and pointers. I created two helper functions (addBeans and removeBeans) to make the code more readable, and to reduce the amount of duplicate code.

For the Player class and it’s derived classes, I didn’t use any special data structures. The SmartPlayer class specifically has two private functions. One is a helper function for the public chooseMove function that does the actual work of finding the best move (the reason for the helper function is so that the function can take more arguments). The other is a near copy of the move function originally created in the Game class (more on that later). The move function is necessary for simulating the possible moves made by both the SmartPlayer and the SmartPlayer’s opponent.

The Game class uses pointers. I didn’t make any helper functions for this class.

**Description of Design for SmartPlayer::chooseMove**

SmartPlayer::chooseMove uses a helper function called SmartPlayer::chooseMoveHelper to determine the optimal move. SmartPlayer::chooseMoveHelper determines the best move through recursion. For every hole that can be played, the helper function will make a move with that hole. If a hole results in a better “play” than the hole that is currently stored, that hole becomes the new “optimal” hole. Determining how good a “play” is is determined when the game is over, the maximum recursion depth is reached, or the maximum time allotted is reached. For a game over condition, if the player loses, that’s the worst possible play. If the player wins, that’s the best possible play. For the max recursion depth or out of time conditions, then the best play is determined based on the difference between the pots. For the south player, the value of South pot minus the value of North pot should be as large as possible (the larger, the better the play). For the north player, the difference should be as small as possible (the smaller the better the play).

**Pseudocode for non-trivial Board functions**

bool Board::sow(Side s, int hole, Side& endSide, int& endHole)

{

if the hole is invalid

return false

make a copy of s and hole to represent where the we are on the board

remove the beans from the initial hole

while there are still beans left to sow

change to the next hole number

skip the opponent’s pot if necessary

add 1 bean to the current hole

subtract one bean from what we have

if we are at a pot switch sides

if we ended on a pot, switch the sides again

assign the side we ended on to endSide

assign the hole we ended on to endHole

return true

}

**Pseudocode for non-trivial Player functions**

int HumanPlayer::chooseMove(const Board& b, Side s) const

{

if a move can’t be made

return -1

ask for hole input

while the inputted hole is invalid

keep asking for a valid hole

return the inputted hole

}

int BadPlayer::chooseMove(const Board& b, Side s) const

{

if a move can’t be made

return -1

for every hole on the player’s side

if there are beans in the hole

return this hole

}

int SmartPlayer::chooseMove(const Board& b, Side s) const

{

if a move can’t be made

return -1

create optimal hole and difference variables as well as a timer

chooseMoveHelper(alarm clock, board, side, optHole, diff, max recursive depth)

if no optimal hole found

for every hole on the player’s side

if there are beans in the hole

return this hole

return the optimal hole

}

void SmartPlayer::chooseMoveHelper(AlarmClock& ac, Board b, Side s, int& optHole, int& diff, int depth) const

{

initialize the diff variable based on what side it is

diff is a very small value if the side is South

diff is a very large value is the side is North

//Base cases

if the game is over

if there is a tie

optHole = -1

diff = 0

return

if south is the winner

optHole = -1

diff = a very big number

return

if north is the winner

optHole = -1

diff = a very small number

return

if we are at the max recursion depth

optHole = -1

diff = value of South pot - value of North pot

return

if we are out of time

optHole = -1

diff = value of South pot - value of North pot

return

//Recursive steps

determine possible holes that can move and store in a dynamically allocated array

for each possible hole (from the dynam. array)

create a copy of the board

bool moveAgain will determine if the same side will move

do a move from the hole (move is implemented as a helper function)

move will assign true or false to moveAgain

create variables diff2 and optHole2

if the same side moves again, call chooseMoveHelper with the same side, with the copy of the board, and decrement the depth – also use diff2 and optHole2

if the other side moves, call chooseMoveHelper with the opponent’s side, the copy of the board, and decrement depth - also use diff2 and optHole2

if diff2 is greater than diff1 and it is South’s turn

optHole is now this hole

diff is now diff2

if diff2 is less than diff1 and it is North’s turn

optHole is now this hole

diff is now diff2

delete the dynamically allocated array

return

}

bool SmartPlayer::move(Board& b, Side s, int hole, bool& moveAgain) const (helper function)

{

if the game is over, return false

sow from the hole and side inputted

If we end on the opponent’s side

if the game is over

sweep the board

return true

if the hole we end on is not a pot, has one bean, and the number of beans on the opposite side is not 0

capture (move beans from both sides to the player’s pot)

if the player ends on his hole

moveAgain is true

if the game is over

sweep the board

return true

}

**Pseudocode for non-trivial Game functions**

void Game::status(bool& over, bool& hasWinner, Side& winner) const

{

if the game is not over

over = false

return

over = true

assume there is a winner

if there is a tie, winner = false

if there is a winner, determine who is the winner (based on who has the most beans in their pot)

}

bool Game::move()

{

if the game is over, return false

pointer currentPlayer points to the player who’s turn it is

bool playerTurn determines if a player gets to play another hole in the same turn

while playerTurn is true

determine hole to sow using chooseMove function of currentPlayer

if hole returned is -1

return false

sow from the hole and side inputted

If we end on the opponent’s side

break out of loop

if the hole we end on is not a pot, has one bean, and the number of beans on the opposite side is not 0

capture (move beans from both sides to the player’s pot)

playerTurn is false

if the player ends on his hole

playerTurn is true

display the board

change the status of who’s turn it is

if the game is over

display the board

sweep the board

return true

}

void Game::play()

{

display initial board

while the game is not over

display the name of the player who’s turn it is

if both players aren’t interactive

require an ENTER keystroke

move()

display the board

determine the status of the game

determine who is the winner and print their name (Tie if there is no winner)

}

**Known Bugs, Serious Inefficiencies, or Notable Problems**

The biggest problem I had was implementing the SmartPlayer::chooseMove function. I tried coming up with my own design of the implementation, but after several failed attempts I looked at the pseudocode given by Professor Smallberg, implemented it, and that worked perfectly. So far, all bugs I have come across have been fixed. Obviously there mayh be bugs still lurking around, but most of them hopefully have been squashed.

**List of Test Cases**

Board Tests

|  |
| --- |
| Board b(3, 2); *//normal board*  assert(b.holes() == 3 && b.totalBeans() == 12 &&  b.beans(SOUTH, POT) == 0 && b.beansInPlay(SOUTH) == 6);  b.setBeans(SOUTH, 1, 1); *//normal setBeans*  b.moveToPot(SOUTH, 2, SOUTH); *//normal moveToPot*  assert(b.totalBeans() == 11 && b.beans(SOUTH, 1) == 1 &&  b.beans(SOUTH, 2) == 0 && b.beans(SOUTH, POT) == 2 &&  b.beansInPlay(SOUTH) == 3);  Side es;  **int** eh;  b.sow(SOUTH, 3, es, eh);  *//check a normal valid sow*  assert(es == NORTH && eh == 3 && b.beans(SOUTH, 3) == 0 &&  b.beans(NORTH, 3) == 3 && b.beans(SOUTH, POT) == 3 &&  b.beansInPlay(SOUTH) == 1 && b.beansInPlay(NORTH) == 7);  *//Board with 0 for both parameters*  Board c(0,0);  assert(c.holes() == 1 && c.totalBeans() == 0 &&  c.beans(SOUTH, POT) == 0 && c.beansInPlay(SOUTH) == 0);  *//Board with negative values for both parameters*  Board d(-1,-1);  assert(d.holes() == 1 && d.totalBeans() == 0 &&  d.beans(SOUTH, POT) == 0 && d.beansInPlay(SOUTH) == 0);  *//check copy constructor*  Board bb(b);  assert(es == NORTH && eh == 3 && b.beans(SOUTH, 3) == 0 &&  b.beans(NORTH, 3) == 3 && b.beans(SOUTH, POT) == 3 &&  b.beansInPlay(SOUTH) == 1 && b.beansInPlay(NORTH) == 7);  assert(!bb.setBeans(SOUTH, 1, -1)); *//invalid setBeans*  assert(!bb.setBeans(SOUTH, 4, 1)); *//invalid setBeans*  assert(!bb.moveToPot(SOUTH, 0, NORTH)); *//invalid moveToPot*  assert(!bb.sow(SOUTH, 0, es, eh)); *//invalid sow*  assert(!bb.sow(SOUTH, -1, es, eh)); *//invalid sow*  *//check that previous operations had no impact on the board*  assert(es == NORTH && eh == 3 && b.beans(SOUTH, 3) == 0 &&  b.beans(NORTH, 3) == 3 && b.beans(SOUTH, POT) == 3 &&  b.beansInPlay(SOUTH) == 1 && b.beansInPlay(NORTH) == 7); |

Player Tests

|  |
| --- |
| **void** doGameTests()  {  *//check a game that forces the any to move in a specific sequence of steps*  BadPlayer bp1("Bart");  BadPlayer bp2("Homer");  Board b(3, 0);  b.setBeans(SOUTH, 1, 2);  b.setBeans(NORTH, 2, 1);  b.setBeans(NORTH, 3, 2);  Game g(b, &bp1, &bp2);  **bool** over;  **bool** hasWinner;  Side winner;  g.display();  *// Homer*  *// 0 1 2*  *// 0 0*  *// 2 0 0*  *// Bart*  g.status(over, hasWinner, winner);  assert(!over && g.beans(NORTH, POT) == 0 && g.beans(SOUTH, POT) == 0 &&  g.beans(NORTH, 1) == 0 && g.beans(NORTH, 2) == 1 && g.beans(NORTH, 3) == 2 &&  g.beans(SOUTH, 1) == 2 && g.beans(SOUTH, 2) == 0 && g.beans(SOUTH, 3) == 0);    g.move();  g.display();  *// 0 1 0*  *// 0 3*  *// 0 1 0*  g.status(over, hasWinner, winner);  assert(!over && g.beans(NORTH, POT) == 0 && g.beans(SOUTH, POT) == 3 &&  g.beans(NORTH, 1) == 0 && g.beans(NORTH, 2) == 1 && g.beans(NORTH, 3) == 0 &&  g.beans(SOUTH, 1) == 0 && g.beans(SOUTH, 2) == 1 && g.beans(SOUTH, 3) == 0);    g.move();  g.display();  *// 1 0 0*  *// 0 3*  *// 0 1 0*  g.status(over, hasWinner, winner);  assert(!over && g.beans(NORTH, POT) == 0 && g.beans(SOUTH, POT) == 3 &&  g.beans(NORTH, 1) == 1 && g.beans(NORTH, 2) == 0 && g.beans(NORTH, 3) == 0 &&  g.beans(SOUTH, 1) == 0 && g.beans(SOUTH, 2) == 1 && g.beans(SOUTH, 3) == 0);    g.move();  g.display();  *// 1 0 0*  *// 0 3*  *// 0 0 1*  g.status(over, hasWinner, winner);  assert(!over && g.beans(NORTH, POT) == 0 && g.beans(SOUTH, POT) == 3 &&  g.beans(NORTH, 1) == 1 && g.beans(NORTH, 2) == 0 && g.beans(NORTH, 3) == 0 &&  g.beans(SOUTH, 1) == 0 && g.beans(SOUTH, 2) == 0 && g.beans(SOUTH, 3) == 1);    g.move();  *// 0 0 0*  *// 1 4*  *// 0 0 0*  g.status(over, hasWinner, winner);  assert(over);  assert(over && g.beans(NORTH, POT) == 1 && g.beans(SOUTH, POT) == 4 &&  g.beans(NORTH, 1) == 0 && g.beans(NORTH, 2) == 0 && g.beans(NORTH, 3) == 0 &&  g.beans(SOUTH, 1) == 0 && g.beans(SOUTH, 2) == 0 && g.beans(SOUTH, 3) == 0);  assert(hasWinner && winner == SOUTH);  }  **void** gameTests()  {  *//check a game that is based off the game tree that Prof. Smallberg provided*  SmartPlayer bp1("Bart");  BadPlayer bp2("Homer");    Board b(6, 0);  b.setBeans(NORTH, 2, 1);  b.setBeans(NORTH, 5, 2);  b.setBeans(SOUTH, 3, 2);  b.setBeans(SOUTH, 5, 1);  b.setBeans(NORTH, 0, 22);  b.setBeans(SOUTH, 0, 20);  Game g(b, &bp1, &bp2);  g.play();    }  **void** gameTests1()  {  *//check two games played on the same board where the smart player is South in the first game and North in the second*  Board b(6, 4);  cout << " ====SMART BART====" <<endl;  SmartPlayer bp1("Bart");  BadPlayer bp2("Homer");  Game g(b, &bp1, &bp2);  g.play();    cout << " ====SMART HOMER====" <<endl;  BadPlayer bp11("Bart");  SmartPlayer bp22("Homer");  Game gg(b, &bp11, &bp22);  gg.play();    } |
| void gameTests2()  {  *//game created so that*  HumanPlayer bp1("Bart");  HumanPlayer bp2("Homer");    Board b(6, 0);  b.setBeans(NORTH, 2, 2);  b.setBeans(NORTH, 4, 1);  b.setBeans(SOUTH, 5, 2);  b.setBeans(SOUTH, 3, 1);    Game g(b, &bp1, &bp2);  g.play();  } |

gameTests1 play: checking to see that SmartPlayer will win on the same initial board layout regardless of South or North starting position (when playing against a BadPlayer)

|  |
| --- |
| **====SMART BART====**  **Here is your initial board setup.**  **Homer**  **4 4 4 4 4 4**  **0 0**  **4 4 4 4 4 4**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 6.**  **Homer**  **4 4 4 5 5 5**  **0 1**  **4 4 4 4 4 0**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 1.**  **Homer**  **0 4 4 5 5 5**  **1 1**  **5 5 5 4 4 0**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 5.**  **Homer**  **0 4 4 5 6 6**  **1 2**  **5 5 5 4 0 1**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 2.**  **Homer**  **1 0 4 5 6 6**  **2 2**  **6 6 5 4 0 1**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 1.**  **Homer**  **1 0 4 5 6 6**  **2 3**  **0 7 6 5 1 2**  **Bart**  **Bart chooses hole 2.**  **Homer**  **1 0 4 5 7 7**  **2 4**  **0 0 7 6 2 3**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 1.**  **Homer**  **0 0 4 5 7 7**  **3 4**  **0 0 7 6 2 3**  **Bart**  **Homer chooses hole 3.**  **Homer**  **1 1 0 5 7 7**  **4 4**  **1 0 7 6 2 3**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 6.**  **Homer**  **1 1 0 5 8 8**  **4 5**  **1 0 7 6 2 0**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 1.**  **Homer**  **0 1 0 5 8 8**  **5 5**  **1 0 7 6 2 0**  **Bart**  **Homer chooses hole 2.**  **Homer**  **0 0 0 5 8 8**  **7 5**  **0 0 7 6 2 0**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 3.**  **Homer**  **0 0 0 6 9 9**  **7 6**  **0 0 0 7 3 1**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 4.**  **Homer**  **1 1 1 0 9 9**  **8 6**  **1 1 0 7 3 1**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 2.**  **Homer**  **1 1 0 0 9 9**  **8 8**  **1 0 0 7 3 1**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 1.**  **Homer**  **0 1 0 0 9 9**  **9 8**  **1 0 0 7 3 1**  **Bart**  **Homer chooses hole 2.**  **Homer**  **0 0 0 0 9 9**  **11 8**  **0 0 0 7 3 1**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 5.**  **Homer**  **0 0 0 0 9 10**  **11 9**  **0 0 0 7 0 2**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 5.**  **Homer**  **1 1 1 1 0 10**  **12 9**  **1 1 1 8 0 2**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 6.**  **Homer**  **1 1 1 1 0 11**  **12 10**  **1 1 1 8 0 0**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 1.**  **Homer**  **0 1 1 1 0 11**  **13 10**  **1 1 1 8 0 0**  **Bart**  **Homer chooses hole 2.**  **Homer**  **0 0 1 1 0 11**  **15 10**  **0 1 1 8 0 0**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 4.**  **Homer**  **0 1 2 2 1 12**  **15 11**  **0 1 1 0 1 1**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 2.**  **Homer**  **1 0 2 2 1 12**  **15 11**  **0 1 1 0 1 1**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 3.**  **Homer**  **1 0 2 0 1 12**  **15 14**  **0 1 0 0 1 1**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 1.**  **Homer**  **0 0 2 0 1 12**  **16 14**  **0 1 0 0 1 1**  **Bart**  **Homer chooses hole 3.**  **Homer**  **1 1 0 0 1 12**  **16 14**  **0 1 0 0 1 1**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 6.**  **Homer**  **1 1 0 0 1 12**  **16 15**  **0 1 0 0 1 0**  **Bart**  **Bart chooses hole 5.**  **Homer**  **1 1 0 0 1 0**  **16 28**  **0 1 0 0 0 0**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 1.**  **Homer**  **0 1 0 0 1 0**  **17 28**  **0 1 0 0 0 0**  **Bart**  **Homer chooses hole 2.**  **Homer**  **1 0 0 0 1 0**  **17 28**  **0 1 0 0 0 0**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 2.**  **Homer**  **1 0 0 0 1 0**  **17 28**  **0 0 1 0 0 0**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 1.**  **Homer**  **0 0 0 0 1 0**  **18 28**  **0 0 1 0 0 0**  **Bart**  **Homer chooses hole 5.**  **Homer**  **0 0 0 1 0 0**  **18 28**  **0 0 1 0 0 0**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 3.**  **Homer**  **0 0 0 0 0 0**  **18 30**  **0 0 0 0 0 0**  **Bart**  **Sweep.**  **Homer**  **0 0 0 0 0 0**  **18 30**  **0 0 0 0 0 0**  **Bart**  **The winner is: Bart**  **====SMART HOMER====**  **Here is your initial board setup.**  **Homer**  **4 4 4 4 4 4**  **0 0**  **4 4 4 4 4 4**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 1.**  **Homer**  **4 4 4 4 4 4**  **0 0**  **0 5 5 5 5 4**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 1.**  **Homer**  **0 4 4 4 4 4**  **1 0**  **1 6 6 5 5 4**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 1.**  **Homer**  **0 4 4 4 4 4**  **1 0**  **0 7 6 5 5 4**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 2.**  **Homer**  **1 0 4 4 4 4**  **2 0**  **1 8 6 5 5 4**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 1.**  **Homer**  **1 0 4 4 4 4**  **2 0**  **0 9 6 5 5 4**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 1.**  **Homer**  **0 0 4 4 4 4**  **3 0**  **0 9 6 5 5 4**  **Bart**  **Homer chooses hole 6.**  **Homer**  **0 0 5 5 5 0**  **13 0**  **0 0 6 5 5 4**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 3.**  **Homer**  **0 0 5 5 6 1**  **13 1**  **0 0 0 6 6 5**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 3.**  **Homer**  **1 1 0 5 6 1**  **14 1**  **1 1 0 6 6 5**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 1.**  **Homer**  **1 1 0 5 6 1**  **14 1**  **0 2 0 6 6 5**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 1.**  **Homer**  **0 1 0 5 6 1**  **15 1**  **0 2 0 6 6 5**  **Bart**  **Homer chooses hole 5.**  **Homer**  **1 2 1 6 0 1**  **16 1**  **1 2 0 6 6 5**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 1.**  **Homer**  **1 2 1 6 0 1**  **16 1**  **0 3 0 6 6 5**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 1.**  **Homer**  **0 2 1 6 0 1**  **17 1**  **0 3 0 6 6 5**  **Bart**  **Homer chooses hole 2.**  **Homer**  **1 0 1 6 0 1**  **18 1**  **0 3 0 6 6 5**  **Bart**  **Homer chooses hole 1.**  **Homer**  **0 0 1 6 0 1**  **19 1**  **0 3 0 6 6 5**  **Bart**  **Homer chooses hole 6.**  **Homer**  **0 0 1 6 0 0**  **26 1**  **0 3 0 6 0 5**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 2.**  **Homer**  **0 0 1 6 0 0**  **26 1**  **0 0 1 7 1 5**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 4.**  **Homer**  **1 1 2 0 0 0**  **27 1**  **1 1 1 7 1 5**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 1.**  **Homer**  **1 1 2 0 0 0**  **27 1**  **0 2 1 7 1 5**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 1.**  **Homer**  **0 1 2 0 0 0**  **28 1**  **0 2 1 7 1 5**  **Bart**  **Homer chooses hole 3.**  **Homer**  **1 2 0 0 0 0**  **28 1**  **0 2 1 7 1 5**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 2.**  **Homer**  **1 2 0 0 0 0**  **28 1**  **0 0 2 8 1 5**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 2.**  **Homer**  **2 0 0 0 0 0**  **29 1**  **0 0 2 8 1 5**  **Bart**  **Homer chooses hole 1.**  **Homer**  **0 0 0 0 0 0**  **30 1**  **1 0 2 8 1 5**  **Bart**  **Sweep.**  **Homer**  **0 0 0 0 0 0**  **30 18**  **0 0 0 0 0 0**  **Bart**  **The winner is: Homer** |

gameTests2 play: checking for sweep, clear, multiple plays in a turn

|  |
| --- |
| **Here is your initial board setup.**  **Homer**  **0 2 0 1 0 0**  **0 0**  **0 0 1 0 2 0**  **Bart**  **Bart's turn is next.**  **Choose your move:**  **5**  **Bart chooses hole 5.**  **Homer**  **0 2 0 1 0 0**  **0 1**  **0 0 1 0 0 1**  **Bart**  **Choose your move: 3**  **Bart chooses hole 3.**  **Homer**  **0 2 0 0 0 0**  **0 3**  **0 0 0 0 0 1**  **Bart**  **Homer's turn is next.**  **Choose your move: 2**  **Homer chooses hole 2.**  **Homer**  **1 0 0 0 0 0**  **1 3**  **0 0 0 0 0 1**  **Bart**  **Choose your move: 1**  **Homer chooses hole 1.**  **Homer**  **0 0 0 0 0 0**  **2 3**  **0 0 0 0 0 1**  **Bart**  **Sweep.**  **Homer**  **0 0 0 0 0 0**  **2 4**  **0 0 0 0 0 0**  **Bart**  **The winner is: Bart** |

gameTests1 play with 3 holes, 15 beans per hole (not being tested, but checking extreme cases)

|  |
| --- |
| **====SMART BART====**  **Here is your initial board setup.**  **Homer**  **15 15 15**  **0 0**  **15 15 15**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 3.**  **Homer**  **17 17 17**  **0 3**  **17 17 2**  **Bart**  **Bart chooses hole 1.**  **Homer**  **19 19 19**  **0 6**  **2 20 5**  **Bart**  **Bart chooses hole 2.**  **Homer**  **22 22 22**  **0 9**  **5 2 8**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 1.**  **Homer**  **3 25 25**  **4 9**  **8 5 11**  **Bart**  **Homer chooses hole 1.**  **Homer**  **0 25 25**  **5 9**  **9 6 11**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 3.**  **Homer**  **2 27 27**  **5 11**  **10 7 1**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 1.**  **Homer**  **0 27 27**  **6 11**  **11 7 1**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 2.**  **Homer**  **1 0 28**  **6 41**  **12 0 2**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 1.**  **Homer**  **0 0 28**  **7 41**  **12 0 2**  **Bart**  **Homer chooses hole 3.**  **Homer**  **4 4 4**  **11 41**  **16 4 6**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 3.**  **Homer**  **5 5 5**  **11 42**  **17 5 0**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 1.**  **Homer**  **0 5 6**  **12 42**  **18 6 1**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 1.**  **Homer**  **2 7 9**  **12 45**  **2 9 4**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 1.**  **Homer**  **0 7 9**  **13 45**  **3 9 4**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 1.**  **Homer**  **0 7 9**  **13 46**  **0 10 5**  **Bart**  **Bart chooses hole 2.**  **Homer**  **1 8 11**  **13 48**  **1 1 7**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 1.**  **Homer**  **0 8 11**  **14 48**  **1 1 7**  **Bart**  **Homer chooses hole 2.**  **Homer**  **2 1 12**  **15 48**  **2 2 8**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 3.**  **Homer**  **3 2 13**  **15 50**  **3 3 1**  **Bart**  **Bart chooses hole 3.**  **Homer**  **3 2 13**  **15 51**  **3 3 0**  **Bart**  **Bart chooses hole 2.**  **Homer**  **3 2 14**  **15 52**  **3 0 1**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 1.**  **Homer**  **0 2 14**  **16 52**  **4 1 1**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 3.**  **Homer**  **0 2 14**  **16 53**  **4 1 0**  **Bart**  **Bart chooses hole 2.**  **Homer**  **0 2 0**  **16 68**  **4 0 0**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 2.**  **Homer**  **1 0 0**  **17 68**  **4 0 0**  **Bart**  **Homer chooses hole 1.**  **Homer**  **0 0 0**  **18 68**  **4 0 0**  **Bart**  **Sweep.**  **Homer**  **0 0 0**  **18 72**  **0 0 0**  **Bart**  **The winner is: Bart**  **====SMART HOMER====**  **Here is your initial board setup.**  **Homer**  **15 15 15**  **0 0**  **15 15 15**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 1.**  **Homer**  **17 17 17**  **0 2**  **2 18 17**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 1.**  **Homer**  **2 19 19**  **3 2**  **5 21 19**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 1.**  **Homer**  **2 20 20**  **3 3**  **0 22 20**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 1.**  **Homer**  **0 20 20**  **4 3**  **1 22 20**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 1.**  **Homer**  **0 20 20**  **4 3**  **0 23 20**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  \**Homer chooses hole 3.**  **Homer**  **3 23 2**  **7 3**  **3 26 23**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 1.**  **Homer**  **3 23 2**  **7 4**  **0 27 24**  **Bart**  **Bart chooses hole 2.**  **Homer**  **7 27 6**  **7 8**  **4 3 28**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 2.**  **Homer**  **11 3 10**  **11 8**  **8 7 32**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 1.**  **Homer**  **12 4 11**  **11 9**  **1 9 33**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 3.**  **Homer**  **14 6 1**  **13 9**  **3 10 34**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 1.**  **Homer**  **14 6 1**  **13 10**  **0 11 35**  **Bart**  **Bart chooses hole 2.**  **Homer**  **15 8 3**  **13 12**  **1 1 37**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 1.**  **Homer**  **2 10 5**  **16 12**  **3 3 39**  **Bart**  **Homer chooses hole 1.**  **Homer**  **0 10 5**  **17 12**  **4 3 39**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 1.**  **Homer**  **0 10 6**  **17 13**  **0 4 40**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 2.**  **Homer**  **2 1 7**  **19 13**  **2 5 41**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 1.**  **Homer**  **2 1 7**  **19 13**  **0 6 42**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 1.**  **Homer**  **0 1 7**  **20 13**  **1 6 42**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 1.**  **Homer**  **0 1 7**  **20 13**  **0 7 42**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 3.**  **Homer**  **1 2 0**  **65 13**  **1 8 0**  **Bart**  **Bart's turn is next.**  **Press ENTER to continue:**  **Bart chooses hole 1.**  **Homer**  **1 2 0**  **65 13**  **0 9 0**  **Bart**  **Homer's turn is next.**  **Press ENTER to continue:**  **Homer chooses hole 2.**  **Homer**  **2 0 0**  **66 13**  **0 9 0**  **Bart**  **Homer chooses hole 1.**  **Homer**  **0 0 0**  **67 13**  **1 9 0**  **Bart**  **Sweep.**  **Homer**  **0 0 0**  **67 23**  **0 0 0**  **Bart**  **The winner is: Homer**  **Passed all tests** |