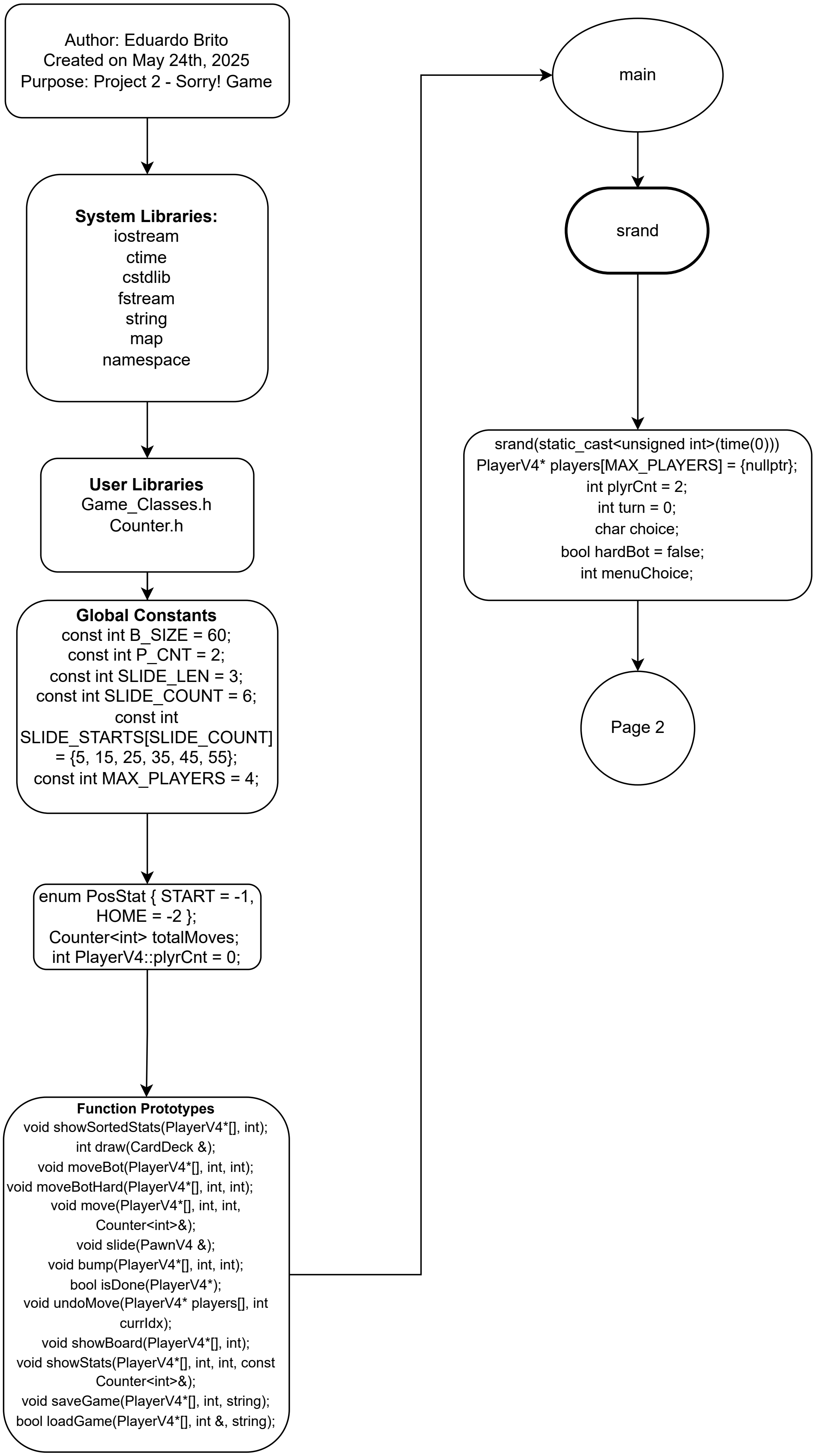


Author: Eduardo Brito
Created on May 24th, 2025
Purpose: Project 2 - Sorry! Game

System Libraries:
iostream
ctime
cstdlib
fstream
string
map
namespace

User Libraries
Game_Classes.h
Counter.h

Global Constants
```
const int B_SIZE = 60;
const int P_CNT = 2;
const int SLIDE_LEN = 3;
const int SLIDE_COUNT = 6;
const int
SLIDE_STARTS[SLIDE_COUNT]
= {5, 15, 25, 35, 45, 55};
const int MAX_PLAYERS = 4;
```

```
enum PosStat { START = -1,
HOME = -2 };
Counter<int> totalMoves;
int PlayerV4::plyrCnt = 0;
```

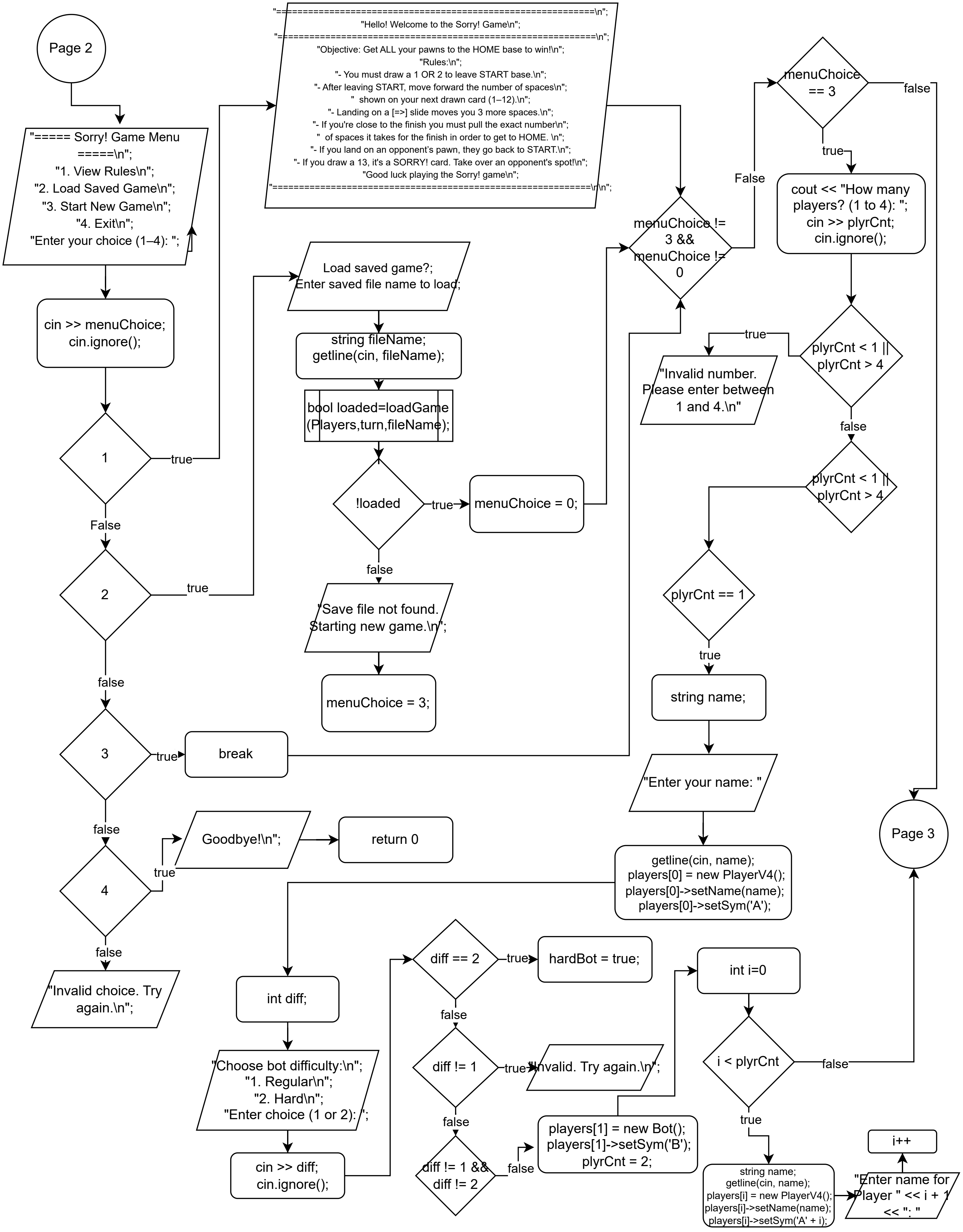
Function Prototypes
```
void showSortedStats(PlayerV4*[], int);
int draw(CardDeck &);
void moveBot(PlayerV4*[], int, int);
void moveBotHard(PlayerV4*[], int, int);
void move(PlayerV4*[], int, int,
Counter<int>&);
void slide(PawnV4 &);
void bump(PlayerV4*[], int, int);
bool isDone(PlayerV4*);
void undoMove(PlayerV4* players[], int
currIdx);
void showBoard(PlayerV4*[], int);
void showStats(PlayerV4*[], int, int, const
Counter<int>&);
void saveGame(PlayerV4*[], int, string);
bool loadGame(PlayerV4*[], int &, string);
```

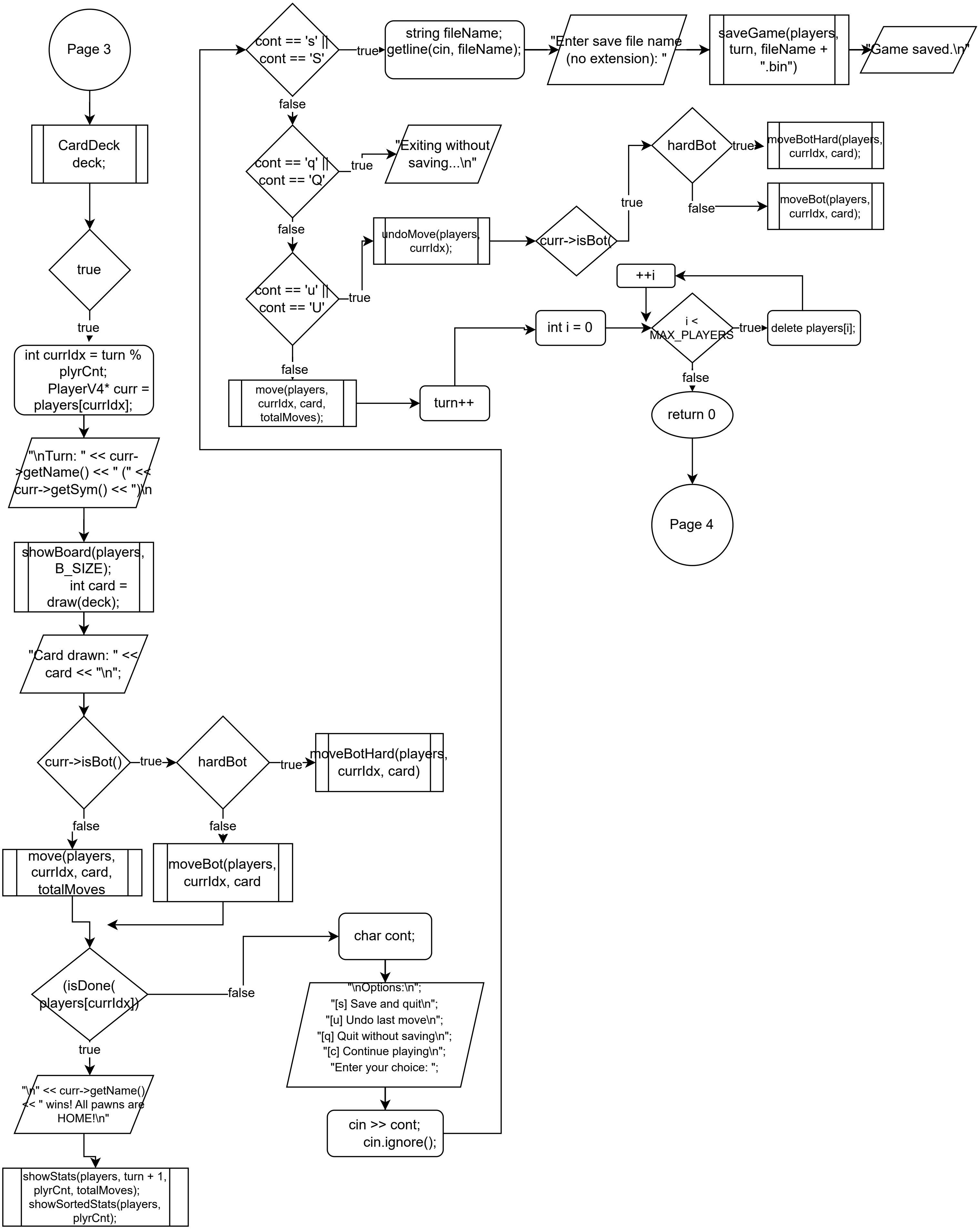
main

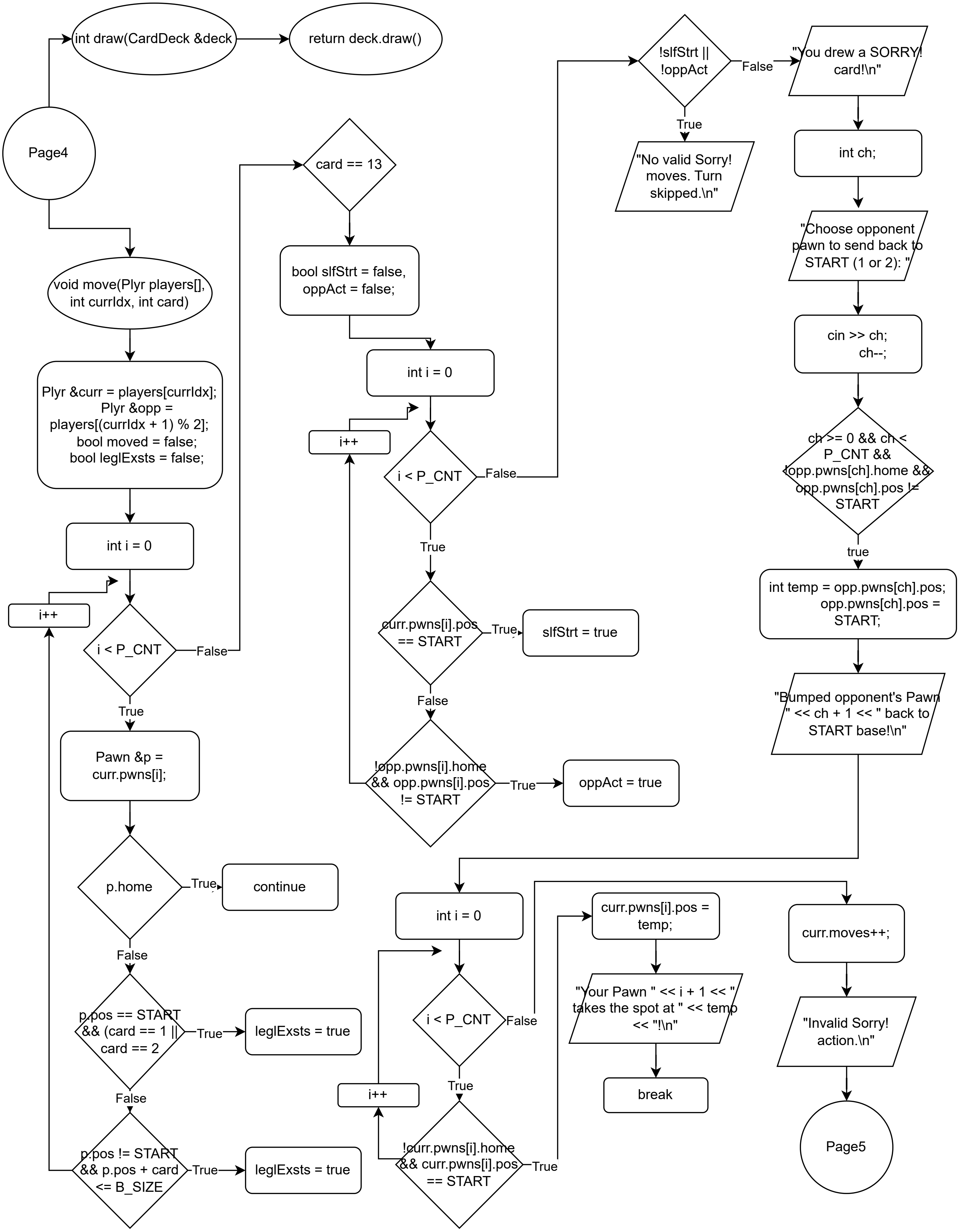
srand

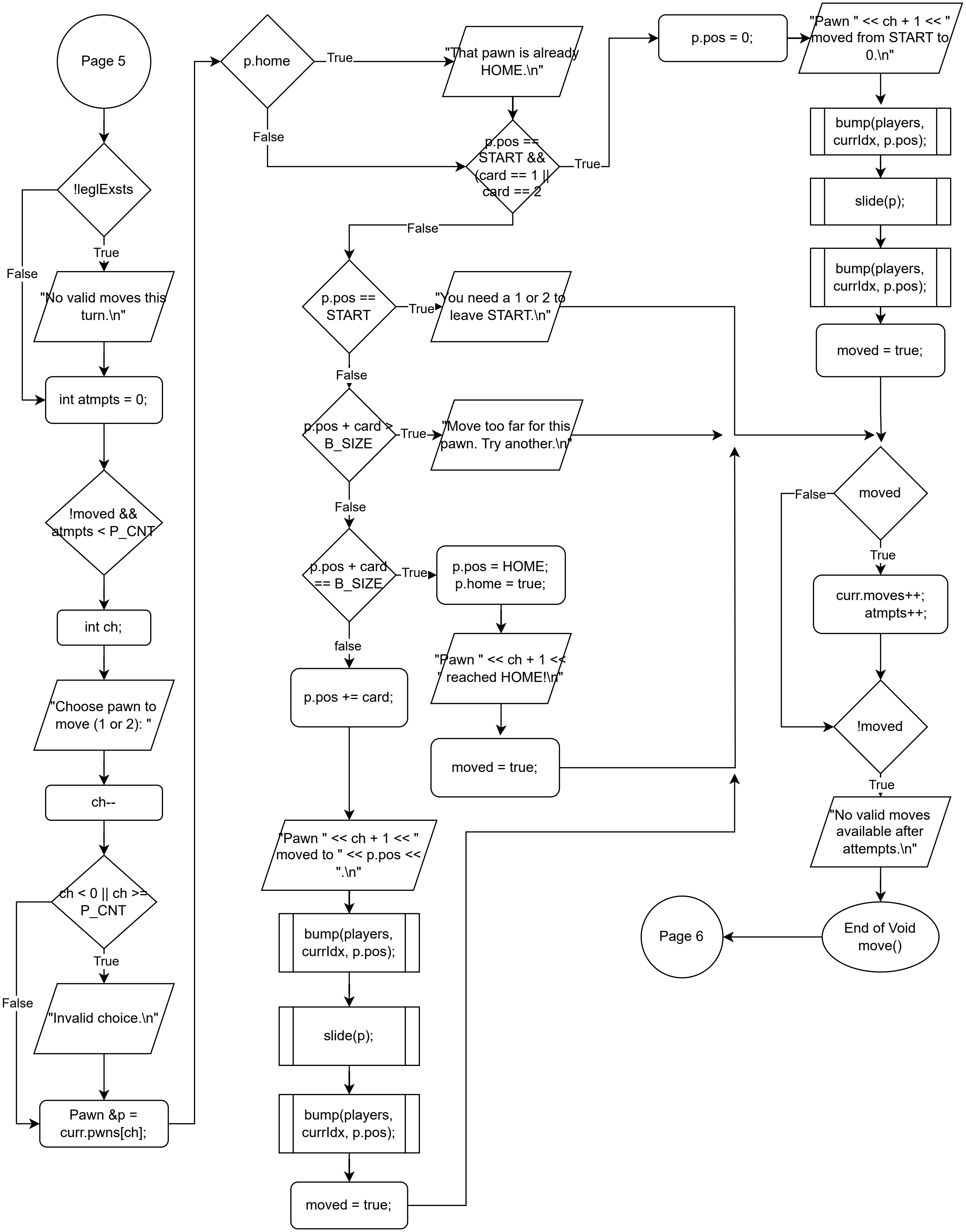
```
srand(static_cast<unsigned int>(time(0)))
PlayerV4* players[MAX_PLAYERS] = {nullptr};
int plyrCnt = 2;
int turn = 0;
char choice;
bool hardBot = false;
int menuChoice;
```

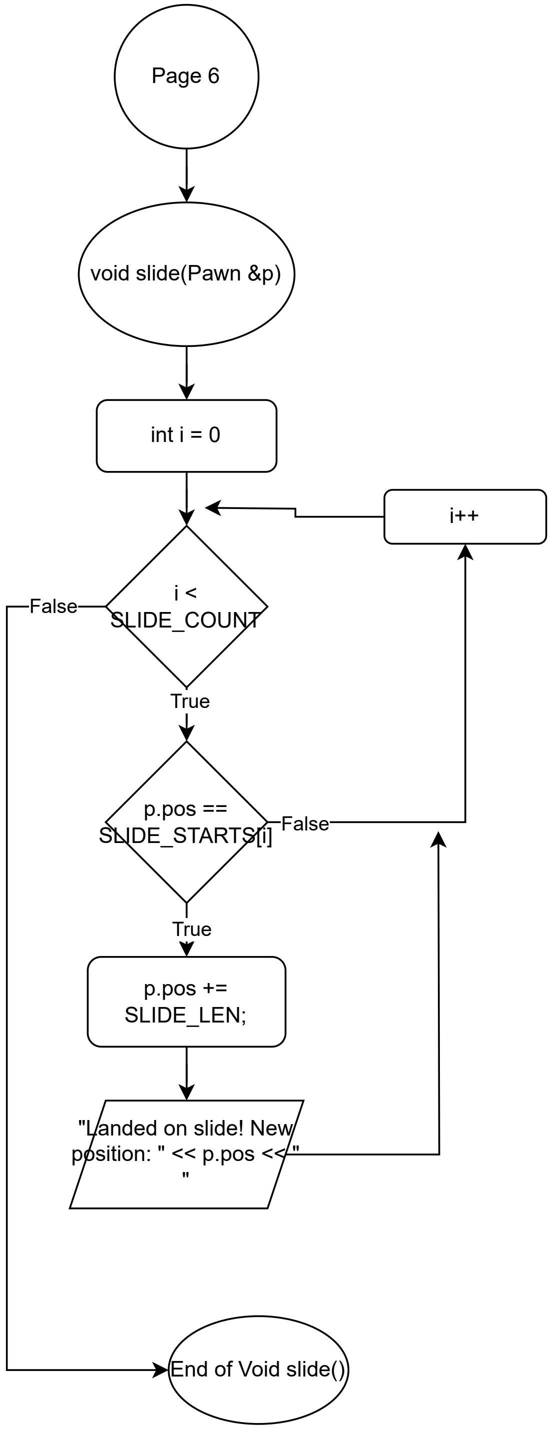
Page 2

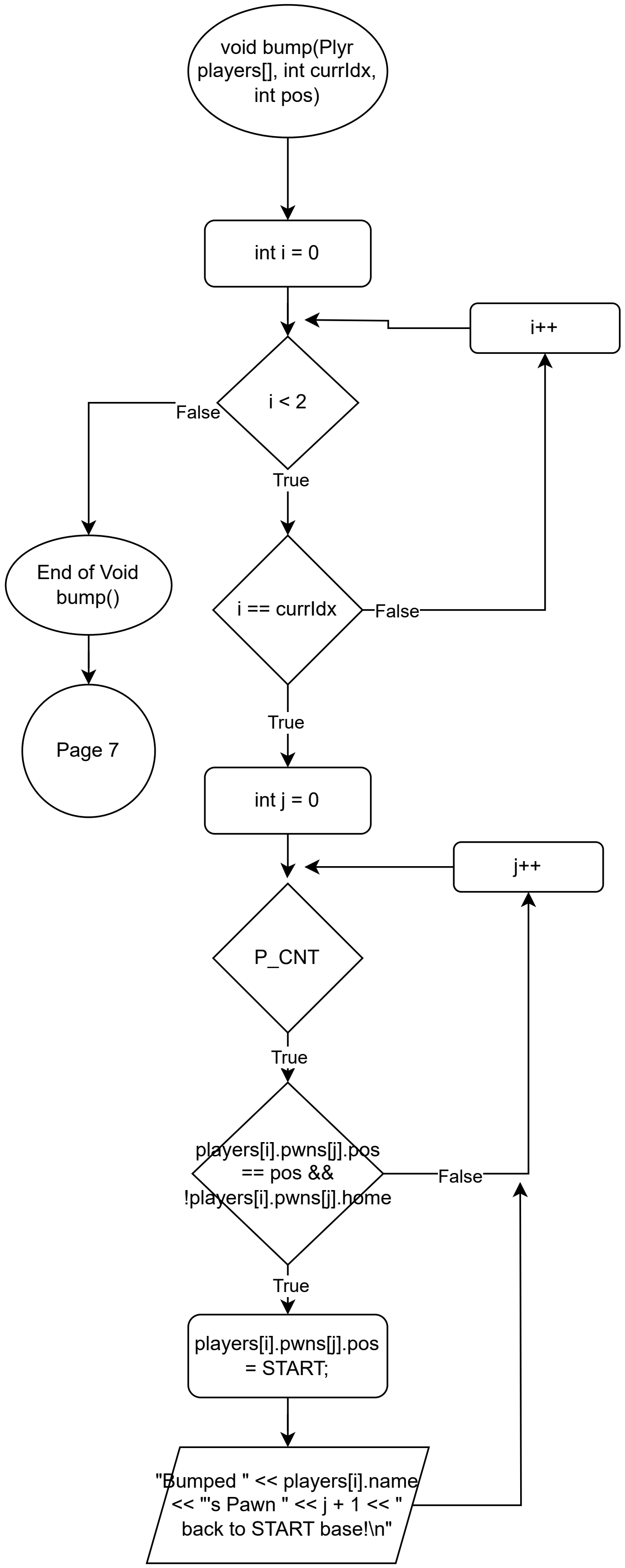

Page 2
"===== Sorry! Game Menu =====\n";
"1. View Rules\n";
"2. Load Saved Game\n";
"3. Start New Game\n";
"4. Exit\n";
"Enter your choice (1–4): ";
cin >> menuChoice;
cin.ignore();
1
true
False
2
true
false
3
true
break
false
4
true
Goodbye!\n";
return 0
false
"Invalid choice. Try again.\n";
"===\n";
"Hello! Welcome to the Sorry! Game\n";
"===\n";
"Objective: Get ALL your pawns to the HOME base to win!\n";
"Rules:\n";
"- You must draw a 1 OR 2 to leave START base.\n";
"- After leaving START, move forward the number of spaces\n";
" shown on your next drawn card (1–12).\n";
"- Landing on a [=>] slide moves you 3 more spaces.\n";
"- If you're close to the finish you must pull the exact number\n";
" of spaces it takes for the finish in order to get to HOME. \n";
"- If you land on an opponent’s pawn, they go back to START.\n";
"- If you draw a 13, it's a SORRY! card. Take over an opponent's spot!\n";
"Good luck playing the Sorry! game\n";
"===\n\n";
Load saved game?;
Enter saved file name to load;
string fileName;
getline(cin, fileName);
bool loaded=loadGame
(Players,turn,fileName);
!loaded
true
menuChoice = 0;
false
"Save file not found.
Starting new game.\n";
menuChoice = 3;
menuChoice != 3 &&
menuChoice != 0
False
menuChoice == 3
false
true
cout << "How many players? (1 to 4): ";
cin >> plyrCnt;
cin.ignore();
plyrCnt < 1 || plyrCnt > 4
true
"Invalid number. Please enter between 1 and 4.\n"
false
plyrCnt < 1 || plyrCnt > 4
plyrCnt == 1
true
string name;
"Enter your name: "
getline(cin, name);
players[0] = new PlayerV4();
players[0]->setName(name);
players[0]->setSym('A');
Page 3
int diff;
"Choose bot difficulty:\n";
"1. Regular\n";
"2. Hard\n";
"Enter choice (1 or 2): ";
cin >> diff;
cin.ignore();
diff == 2
true
hardBot = true;
false
diff != 1
true
"Invalid. Try again.\n";
false
diff != 1 &&
diff != 2
false
players[1] = new Bot();
players[1]->setSym('B');
plyrCnt = 2;
int i=0
i < plyrCnt
false
true
string name;
getline(cin, name);
players[i] = new PlayerV4();
players[i]->setName(name);
players[i]->setSym('A' + i);
"Enter name for Player " << i + 1 << ": "
i++

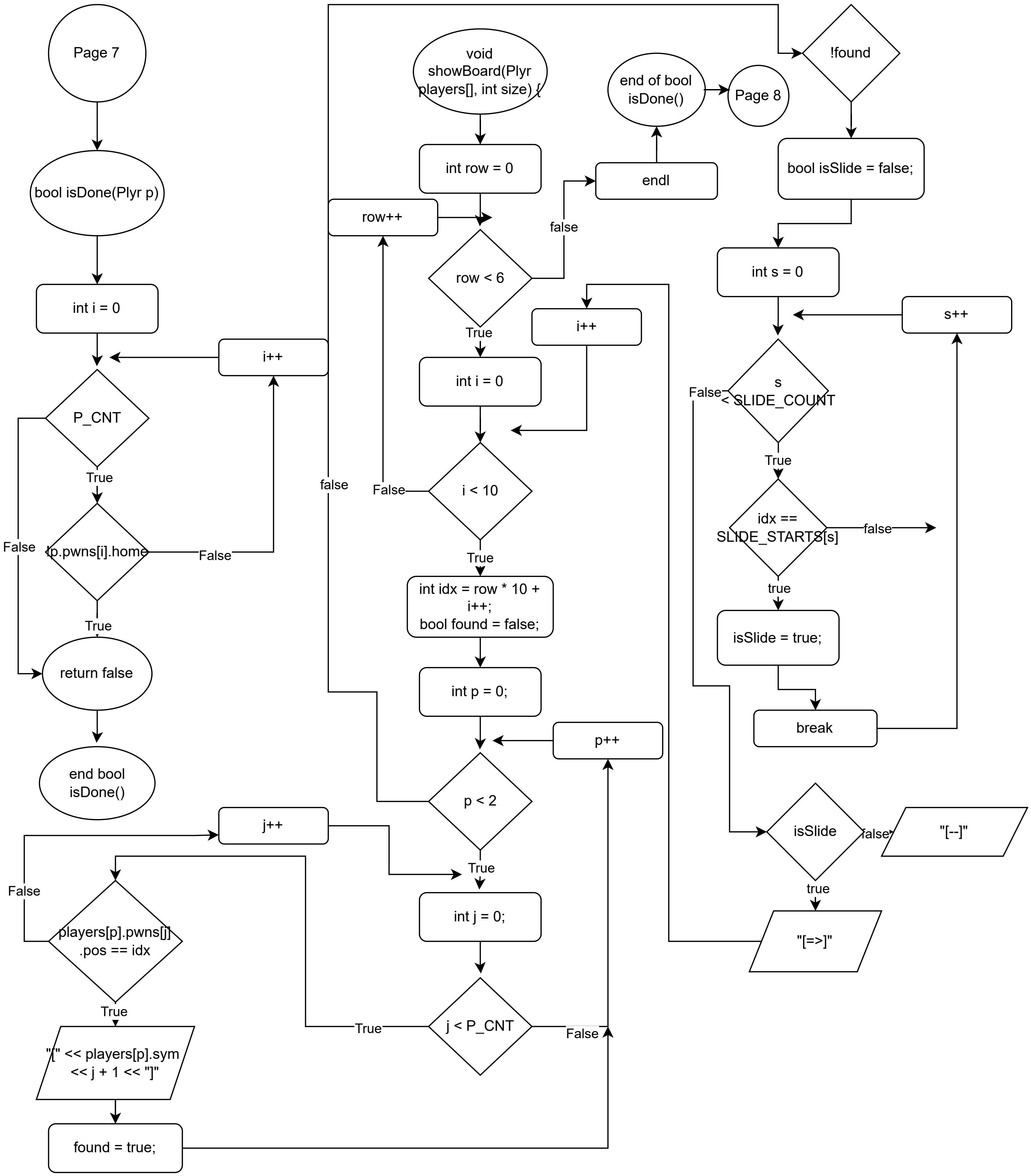

Page 3
CardDeck deck;
true
true
int currIdx = turn % plyrCnt; PlayerV4* curr = players[currIdx];
"\nTurn: " << curr->getName() << " (" << curr->getSym() << ")\n
showBoard(players, B_SIZE); int card = draw(deck);
"Card drawn: " << card << "\n";
curr->isBot()
true
hardBot
true
moveBotHard(players, currIdx, card)
false
move(players, currIdx, card, totalMoves
false
moveBot(players, currIdx, card
(isDone(players[currIdx]))
true
"\n" << curr->getName() << " wins! All pawns are HOME!\n"
showStats(players, turn + 1, plyrCnt, totalMoves); showSortedStats(players, plyrCnt);
false
char cont;
"\nOptions:\n"; "[s] Save and quit\n"; "[u] Undo last move\n"; "[q] Quit without saving\n"; "[c] Continue playing\n"; "Enter your choice: ";
cin >> cont; cin.ignore();
cont == 's' || cont == 'S'
true
string fileName; getline(cin, fileName);
"Enter save file name (no extension): "
saveGame(players, turn, fileName + ".bin")
"Game saved.\n"
false
cont == 'q' || cont == 'Q'
true
"Exiting without saving...\n"
false
cont == 'u' || cont == 'U'
true
undoMove(players, currIdx);
curr->isBot()
true
hardBot
true
moveBotHard(players, currIdx, card);
false
moveBot(players, currIdx, card);
false
move(players, currIdx, card, totalMoves);
turn++
int i = 0
++i
i < MAX_PLAYERS
true
delete players[i];
false
return 0
Page 4

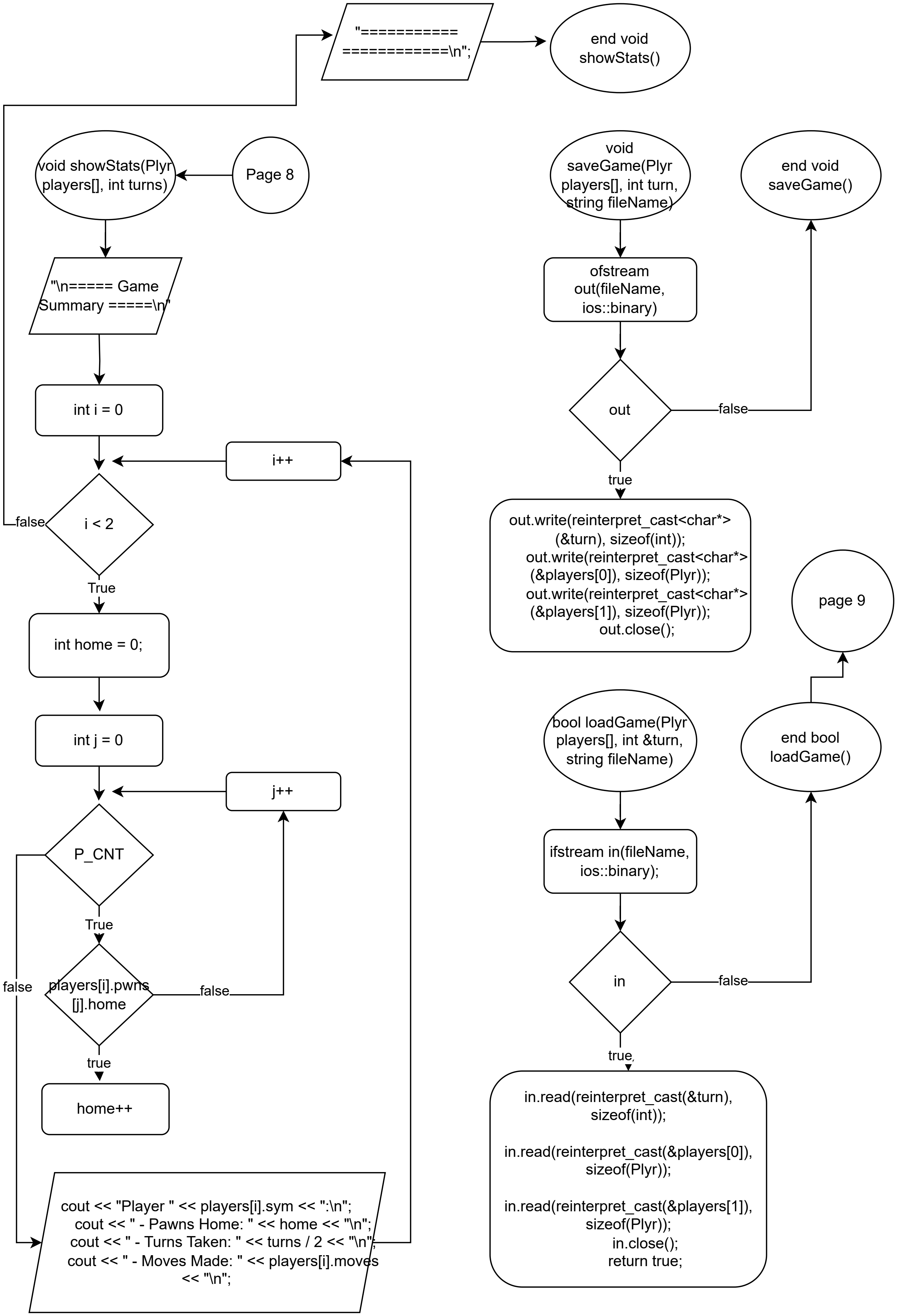

Page4
int draw(CardDeck &deck
return deck.draw()
void move(Plyr players[], int currIdx, int card)
Plyr &curr = players[currIdx]; Plyr &opp = players[(currIdx + 1) % 2]; bool moved = false; bool leglExsts = false;
int i = 0
i++
i < P_CNT
False
True
Pawn &p = curr.pwns[i];
p.home
True
continue
False
p.pos == START && (card == 1 || card == 2
True
leglExsts = true
False
p.pos != START && p.pos + card <= B_SIZE
True
leglExsts = true
card == 13
bool slfStrt = false, oppAct = false;
int i = 0
i++
i < P_CNT
False
True
curr.pwns[i].pos == START
True
slfStrt = true
False
!opp.pwns[i].home && opp.pwns[i].pos != START
True
oppAct = true
!slfStrt || !oppAct
False
True
"No valid Sorry! moves. Turn skipped.\n"
"You drew a SORRY! card!\n"
int ch;
"Choose opponent pawn to send back to START (1 or 2): "
cin >> ch; ch--;
ch >= 0 && ch < P_CNT && !opp.pwns[ch].home && opp.pwns[ch].pos != START
true
int temp = opp.pwns[ch].pos; opp.pwns[ch].pos = START;
"Bumped opponent's Pawn " << ch + 1 << " back to START base!\n"
int i = 0
i++
i < P_CNT
False
True
!curr.pwns[i].home && curr.pwns[i].pos == START
True
curr.pwns[i].pos = temp;
"Your Pawn " << i + 1 << " takes the spot at " << temp << "!\n"
break
curr.moves++;
"Invalid Sorry! action.\n"
Page5

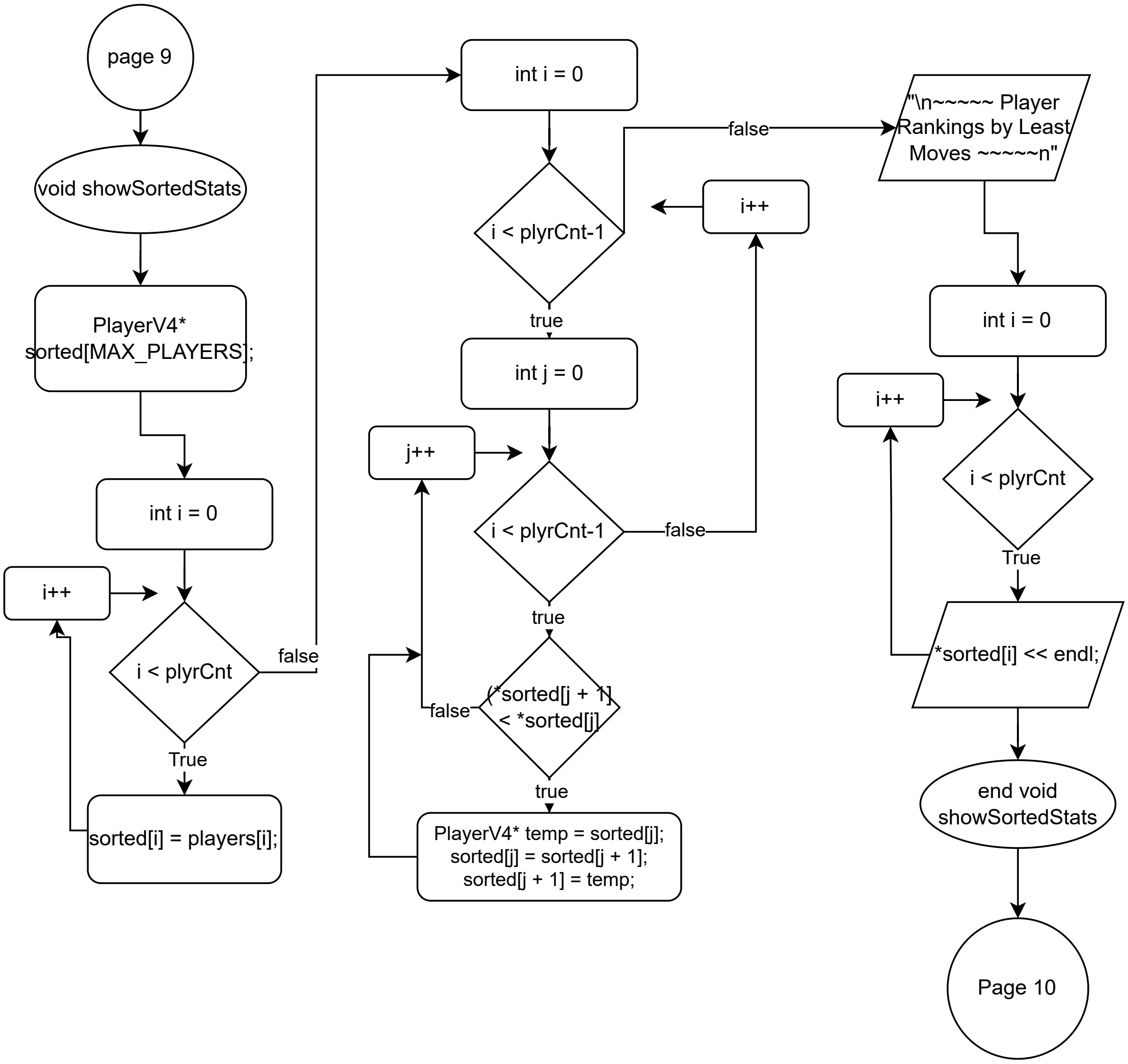

Page 5
!leglExsts
True
False
"No valid moves this turn.\n"
int atmpts = 0;
!moved && atmpts < P_CNT
int ch;
"Choose pawn to move (1 or 2): "
ch--
ch < 0 || ch >= P_CNT
True
False
"Invalid choice.\n"
Pawn &p = curr.pwns[ch];
p.home
True
False
"That pawn is already HOME.\n"
p.pos == START && (card == 1 || card == 2
True
False
p.pos = 0;
"Pawn " << ch + 1 << " moved from START to 0.\n"
bump(players, currIdx, p.pos);
slide(p);
bump(players, currIdx, p.pos);
moved = true;
p.pos == START
True
False
"You need a 1 or 2 to leave START.\n"
p.pos + card > B_SIZE
True
False
"Move too far for this pawn. Try another.\n"
p.pos + card == B_SIZE
True
false
p.pos = HOME; p.home = true;
"Pawn " << ch + 1 << " reached HOME!\n"
moved = true;
p.pos += card;
"Pawn " << ch + 1 << " moved to " << p.pos << ".\n"
bump(players, currIdx, p.pos);
slide(p);
bump(players, currIdx, p.pos);
moved = true;
moved
False
True
curr.moves++; atmpts++;
!moved
True
"No valid moves available after attempts.\n"
End of Void move()
Page 6

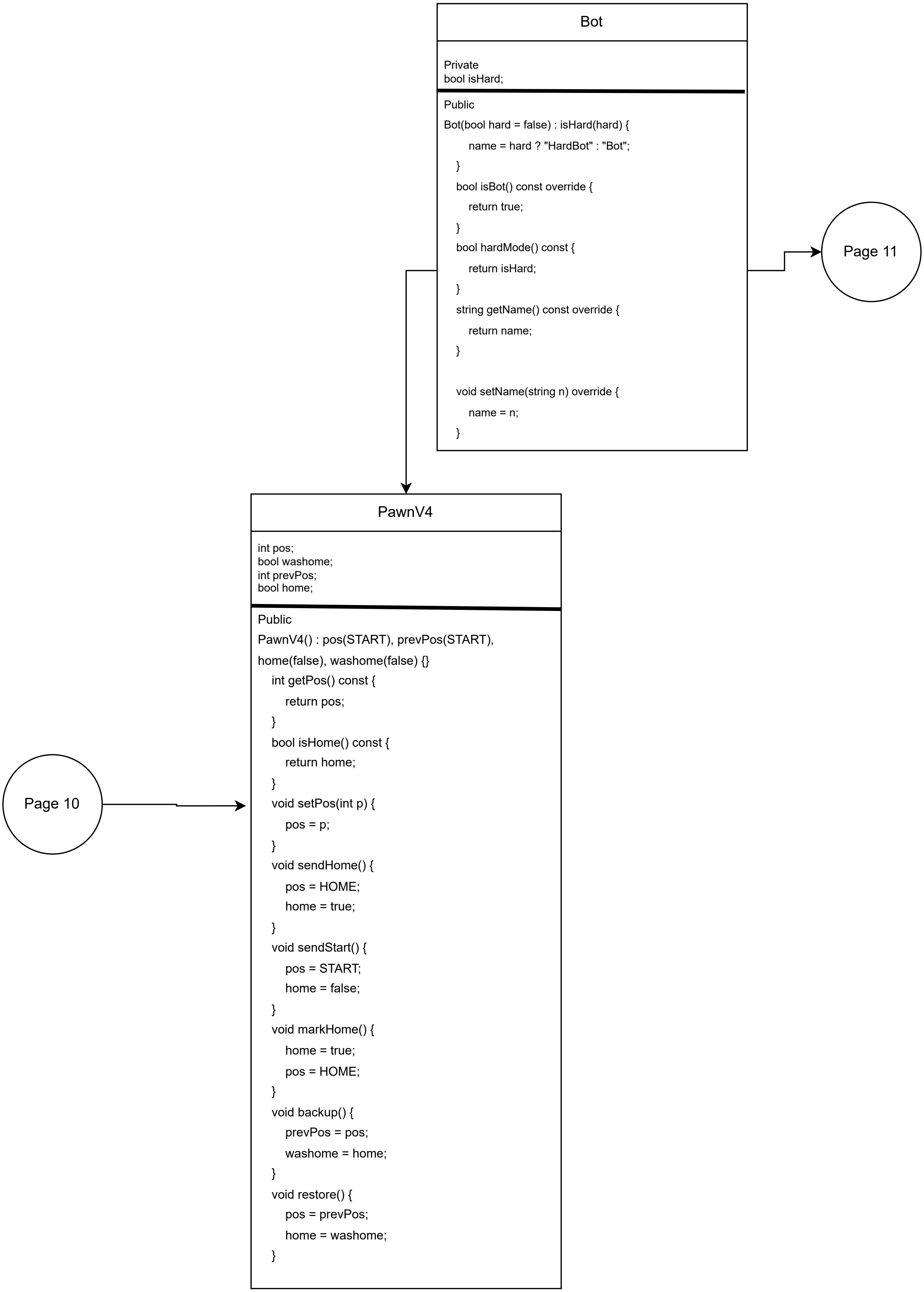

Page 6
void slide(Pawn &p)
int i = 0
i++
i < SLIDE_COUNT
False
True
p.pos == SLIDE_STARTS[i]
False
True
p.pos += SLIDE_LEN;
"Landed on slide! New position: " << p.pos << "
"
End of Void slide()


void bump(Plyr players[], int currIdx, int pos)
int i = 0
i++
i < 2
False
True
End of Void bump()
Page 7
i == currIdx
False
True
int j = 0
j++
P_CNT
True
players[i].pwns[j].pos == pos && !players[i].pwns[j].home
False
True
players[i].pwns[j].pos = START;
"Bumped " << players[i].name << "'s Pawn " << j + 1 << " back to START base!\n"


Page 7
bool isDone(Plyr p)
int i = 0
i++
P_CNT
True
False
!p.pwns[i].home
False
True
return false
end bool isDone()
void showBoard(Plyr players[], int size) {
int row = 0
row++
row < 6
false
True
endl
end of bool isDone()
Page 8
int i = 0
i++
i < 10
False
True
int idx = row * 10 + i++;
bool found = false;
int p = 0;
p++
p < 2
false
True
int j = 0;
j++
j < P_CNT
True
False
players[p].pwns[j].pos == idx
False
True
"[" << players[p].sym << j + 1 << "]"
found = true;
!found
bool isSlide = false;
int s = 0
s++
s < SLIDE_COUNT
False
True
idx == SLIDE_STARTS[s]
false
true
isSlide = true;
break
isSlide
false
true
"[--]"
"[=>]"


"===========
============\n";
end void
showStats()
void showStats(Plyr
players[], int turns)
Page 8
"\n===== Game
Summary =====\n"
int i = 0
i++
i < 2
false
True
int home = 0;
int j = 0
j++
P_CNT
True
false
players[i].pwns
[j].home
false
true
home++
cout << "Player " << players[i].sym << ":\n";
cout << " - Pawns Home: " << home << "\n";
cout << " - Turns Taken: " << turns / 2 << "\n";
cout << " - Moves Made: " << players[i].moves
<< "\n";
void
saveGame(Plyr
players[], int turn,
string fileName)
end void
saveGame()
ofstream
out(fileName,
ios::binary)
out
false
true
out.write(reinterpret_cast<char*>
(&turn), sizeof(int));
out.write(reinterpret_cast<char*>
(&players[0]), sizeof(Plyr));
out.write(reinterpret_cast<char*>
(&players[1]), sizeof(Plyr));
out.close();
page 9
bool loadGame(Plyr
players[], int &turn,
string fileName)
end bool
loadGame()
ifstream in(fileName,
ios::binary);
in
false
true
in.read(reinterpret_cast(&turn),
sizeof(int));
in.read(reinterpret_cast(&players[0]),
sizeof(Plyr));
in.read(reinterpret_cast(&players[1]),
sizeof(Plyr));
in.close();
return true;


page 9
void showSortedStats
PlayerV4* sorted[MAX_PLAYERS];
int i = 0
i++
i < plyrCnt
True
sorted[i] = players[i];
false
int i = 0
i < plyrCnt-1
false
i++
true
int j = 0
j++
i < plyrCnt-1
false
true
(*sorted[j + 1] < *sorted[j]
false
true
PlayerV4* temp = sorted[j];
sorted[j] = sorted[j + 1];
sorted[j + 1] = temp;
"\n~~~~~ Player Rankings by Least Moves ~~~~~n"
int i = 0
i++
i < plyrCnt
True
*sorted[i] << endl;
end void showSortedStats
Page 10



Bot

Private
bool isHard;

Public

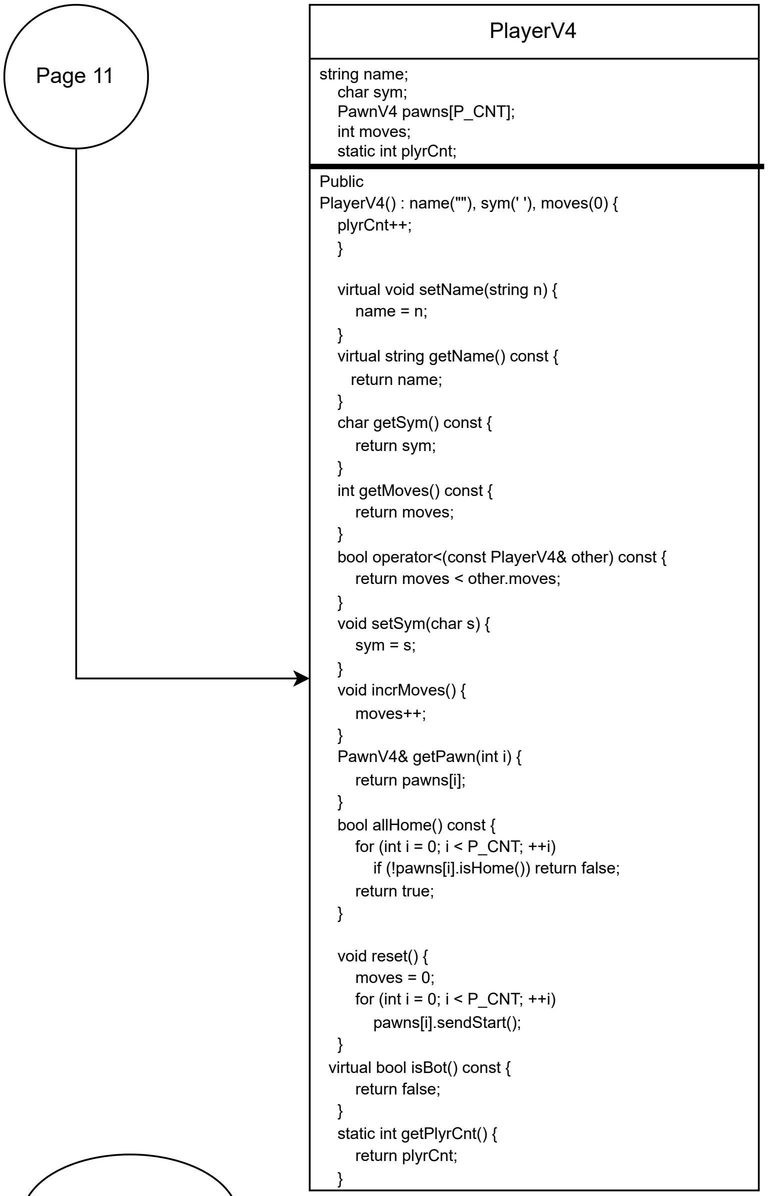
```
Bot(bool hard = false) : isHard(hard) {
      name = hard ? "HardBot" : "Bot";
   }
   bool isBot() const override {
      return true;
   }
   bool hardMode() const {
      return isHard;
   }
   string getName() const override {
      return name;
   }

   void setName(string n) override {
      name = n;
   }
```

PawnV4

```
int pos;
bool washome;
int prevPos;
bool home;
```

Public

```
PawnV4() : pos(START), prevPos(START),
home(false), washome(false) {}
   int getPos() const {
      return pos;
   }
   bool isHome() const {
      return home;
   }
   void setPos(int p) {
      pos = p;
   }
   void sendHome() {
      pos = HOME;
      home = true;
   }
   void sendStart() {
      pos = START;
      home = false;
   }
   void markHome() {
      home = true;
      pos = HOME;
   }
   void backup() {
      prevPos = pos;
      washome = home;
   }
   void restore() {
      pos = prevPos;
      home = washome;
   }
```



PlayerV4

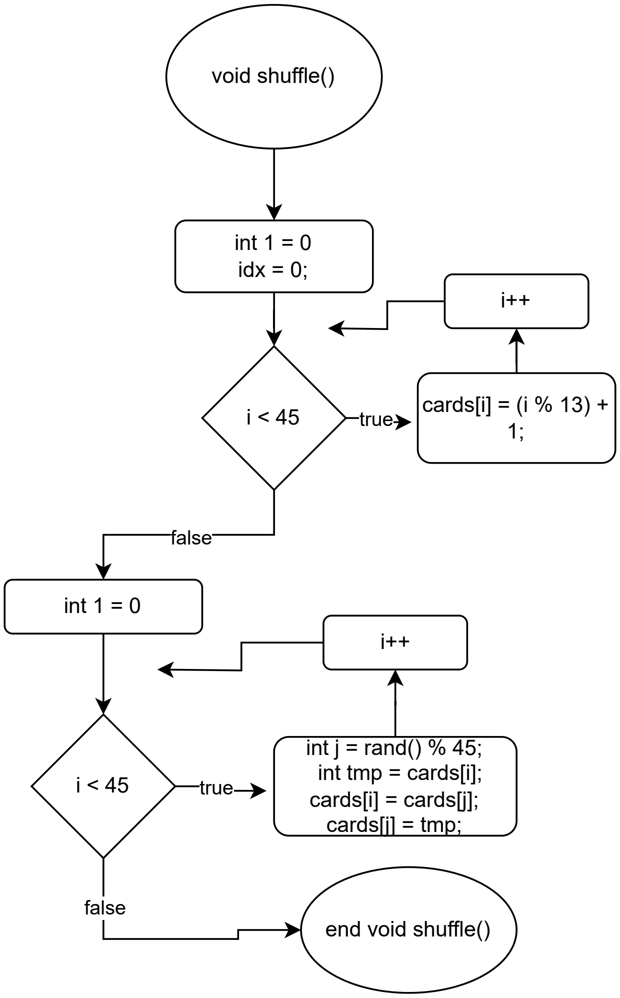
```
string name;
    char sym;
    PawnV4 pawns[P_CNT];
    int moves;
    static int plyrCnt;
```

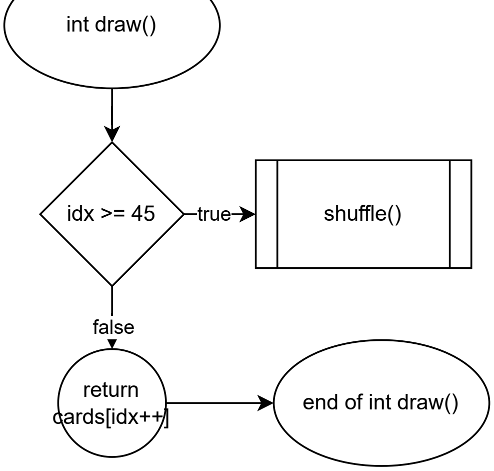
```
Public
PlayerV4() : name(""), sym(' '), moves(0) {
    plyrCnt++;
    }

    virtual void setName(string n) {
        name = n;
    }
    virtual string getName() const {
       return name;
    }
    char getSym() const {
        return sym;
    }
    int getMoves() const {
        return moves;
    }
    bool operator<(const PlayerV4& other) const {
        return moves < other.moves;
    }
    void setSym(char s) {
        sym = s;
    }
    void incrMoves() {
        moves++;
    }
    PawnV4& getPawn(int i) {
        return pawns[i];
    }
    bool allHome() const {
        for (int i = 0; i < P_CNT; ++i)
            if (!pawns[i].isHome()) return false;
        return true;
    }

    void reset() {
        moves = 0;
        for (int i = 0; i < P_CNT; ++i)
            pawns[i].sendStart();
    }
  virtual bool isBot() const {
        return false;
    }
    static int getPlyrCnt() {
        return plyrCnt;
    }
```

CardDeck

```
int cards[45];
    int idx;
```

```
Public
CardDeck() {
        shuffle();
    }
 void shuffle() {

    }
```


void shuffle()
int 1 = 0
idx = 0;
i < 45
true
cards[i] = (i % 13) + 1;
i++
false
int 1 = 0
i < 45
true
int j = rand() % 45;
int tmp = cards[i];
cards[i] = cards[j];
cards[j] = tmp;
i++
false
end void shuffle()


int draw()
idx >= 45
true
shuffle()
false
return cards[idx++]
end of int draw()