CIS 17C Project 1

<Mancala>

CIS 17C-43484
Paul Ingram
05/02/2022

Introduction

What are you coding and why did you choose this game?
For this project I decided to code Mancala. I chose this game because I used to play it all the time when I was younger. I even had a Club Penguin themed mancala board. Another reason is because both of my 17A projects were Mancala and I wanted to see how I would code it differently this time. I think Mancala has simple enough rules that provide a clear plan for developing the code.

How long did you spend, how many lines, classes, etc.?
I spent about a week coding this game. I finished with about 900 lines. The program uses two main classes in the GameBoard class and the SimGame class. The GameBoard class basically holds the players scores, their pits on each side, and each player's play. The class needed two separate functions for each player's choice since one side needed everything done in reverse. The game is played in a counter-clockwise motion so the person on the top needed code that would iterate through their cells in reverse. The SimGame class was able to hold a GameBoard and all that was left to do is to make a bot that would randomly choose a pit.

Where on GitHub is it located?
https://github.com/pingram32/IngramPaul_CIS17C_43484

Approach to Development

Concepts
The two lists which served as the player pits used both a forward and reverse iterator. The game is played in a counterclockwise motion so whoever was on the top of the board needed a reverse iterator that would start from the end of the list and make its way to the beginning. When creating the scoring mechanism I needed to create something that wouldn’t use a random access iterator since this project used lists as the player pits. Trivial iterators allowed me to dereference each iterator when traversing the lists. Because of this, I could mutate the information that was stored in the list and change the number of marbles in each of the following pits from the player's choice. Input iterators were implemented in the bot play mechanism where a random number must be found in a list of possible choices. An output iterator was used for the copying process where the list of possible moves for the bot was copied from a set numbered 1-6.

Version Control
V1 Focused on developing code for one player's choice. I created two functions which checked if the game was over and printed out the game board.

V2 Implemented the go again feature which checked if the last marble of someone's play landed in their score bank. If it did, a boolean value is set and the code knows to give the same player another turn. Still only working on player 1's move.

V3 Main is converted into a Board class where the class holds the players' scores and the information about the pits on either side. This makes for a clean looking main file where the main functions called are for printing the board and the player's turn. The game also has an added function which announces the winner at the end of the game.

V4 Developed the SimGame class which uses the Board object. Changes are made to the original functions where the "Bot's" choices are outputted. a new function for a bot play is created. I had an issue where I played against the bot and when the Bot earned another turn it expected a user to input another choice. I realized I had to rework the play function so that it automatically generated another choice when it earned another turn.

V5 Developed a capture system where if the last marble ends on an empty pit from the side it was played from the cell across from it is captured.

Game Rules

The rules are simple in Mancala. Each side has 6 pits where each pit starts off with 4 marbles. The first player chooses a pit and adds one marble to each of the following pits until they no longer have any marbles left from the pit that they chose from. If the player happens to land their last marble in their score pit (located to the right and left of each side) they receive another turn. If the last marble lands in an empty pit on the same side it was played from the pit across from it is captured and goes to that side's score bank. Once the first player has finished making their move, the second player chooses a cell on their respective side. Neither player may select a cell that is not on their side. The game ends once one side has 0 marbles left in all of the pits. Once the game is finished all the marbles left over are consolidated to their respective side's score pit and the winner is determined by which player has more marbles in their score pit.

Description of Code

Organization
The project is organized into a main file, a Board class, and a SimGame class. Main basically serves as the main menu where the user selects if they would like to do a 1v1 or a 1vBot where the Bot is really just a random number generator. I imagined the Board as an actual mancala board which holds the information on both sides, the player scores, and each player's play function. The SimGame class basically has a Board that instead uses a bot function which returns a random number for player two's input.

Classes
Board, SimGame

Sample Input/Output

```
|6|6|0|6|0|5|
2==========1
|5|5|4|4|4|0|
2's turn: Choose a cell 1-6 LR on bottom row1
Capture!

|6|6|0|6|0|0|
2==========7
|0|6|5|5|5|0|
1's turn: Choose a cell 1-6 LR on top row
```

```
|0|4|4|4|4|4|
1==========1
|5|0|3|6|6|6|
1's turn: Choose a cell 1-6 LR on top row5
Capture!

|0|5|5|5|0|4|
7==========1
|0|0|3|6|6|6|
2's turn: Choose a cell 1-6 LR on bottom row
```

```
|4|4|4|4|4|4|
0==========0
|4|4|4|4|4|4|
1's turn: Choose a cell 1-6 LR on top row4

|5|5|5|0|4|4|
1==========0
|4|4|4|4|4|4|
You have earned another turn, go again
5
```

```
|6|6|6|1|0|4|
1==========0
|4|4|4|4|4|4|
2's turn: Choose a cell 1-6 LR on bottom row3

|6|6|6|1|0|4|
1==========1
|4|4|0|5|5|5|
You have earned another turn, go again
2
```

Checkoff Sheet

1.Container classes (Where in code did you put each of these Concepts and how were they used?

- 1.Sequences (At least 1)
  - 1.list **Line 25 and 26 in "Board.h". Holds the players' side information.**
  - 2.slist
  - 3.bit_vector
- 2.Associative Containers (At least 2)
  - 1.set **Line 18 in "SimGame.h". Holds the set of bot choices.**
  - 2.map **Line 24 in "Board.h". Keeps track of player score with name**
  - 3.hash
- 3.Container adaptors (At least 2)
  - 1.stack **Line 28 in "Board.h". Keeps track of the current player**
  - 2.queue **Line 27 in "Board.h". Keeps track of the game history**
  - 3.priority_queue

2.Iterators

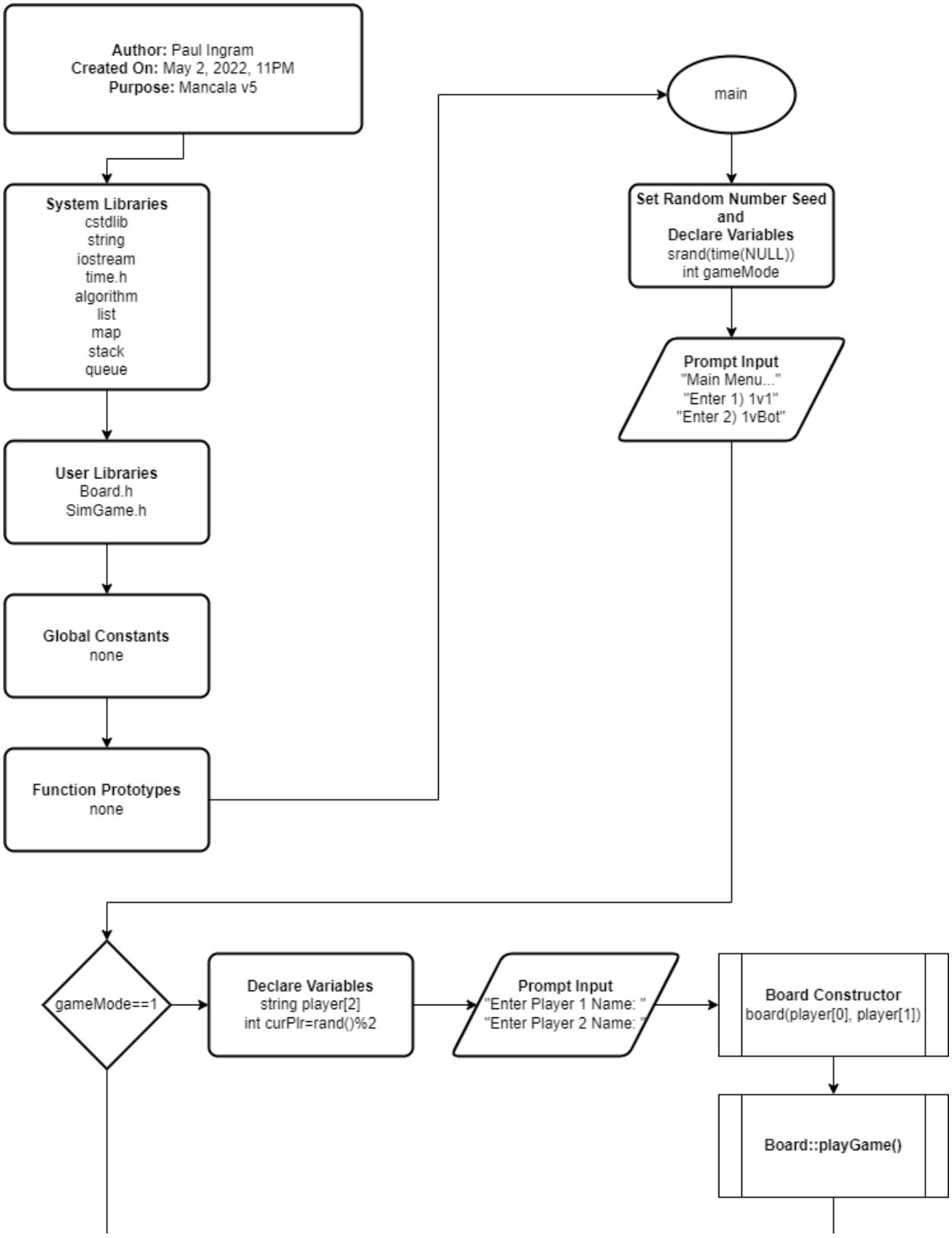
- 1.Concepts (Describe the iterators utilized for each Container)
  - 1.Trivial Iterator **(For 1-6 please refer to the "Concepts" section)**
  - 2.Input Iterator
  - 3.Output Iterator
  - 4.Forward Iterator
  - 5.Bidirectional Iterator
  - 6.Random Access Iterator

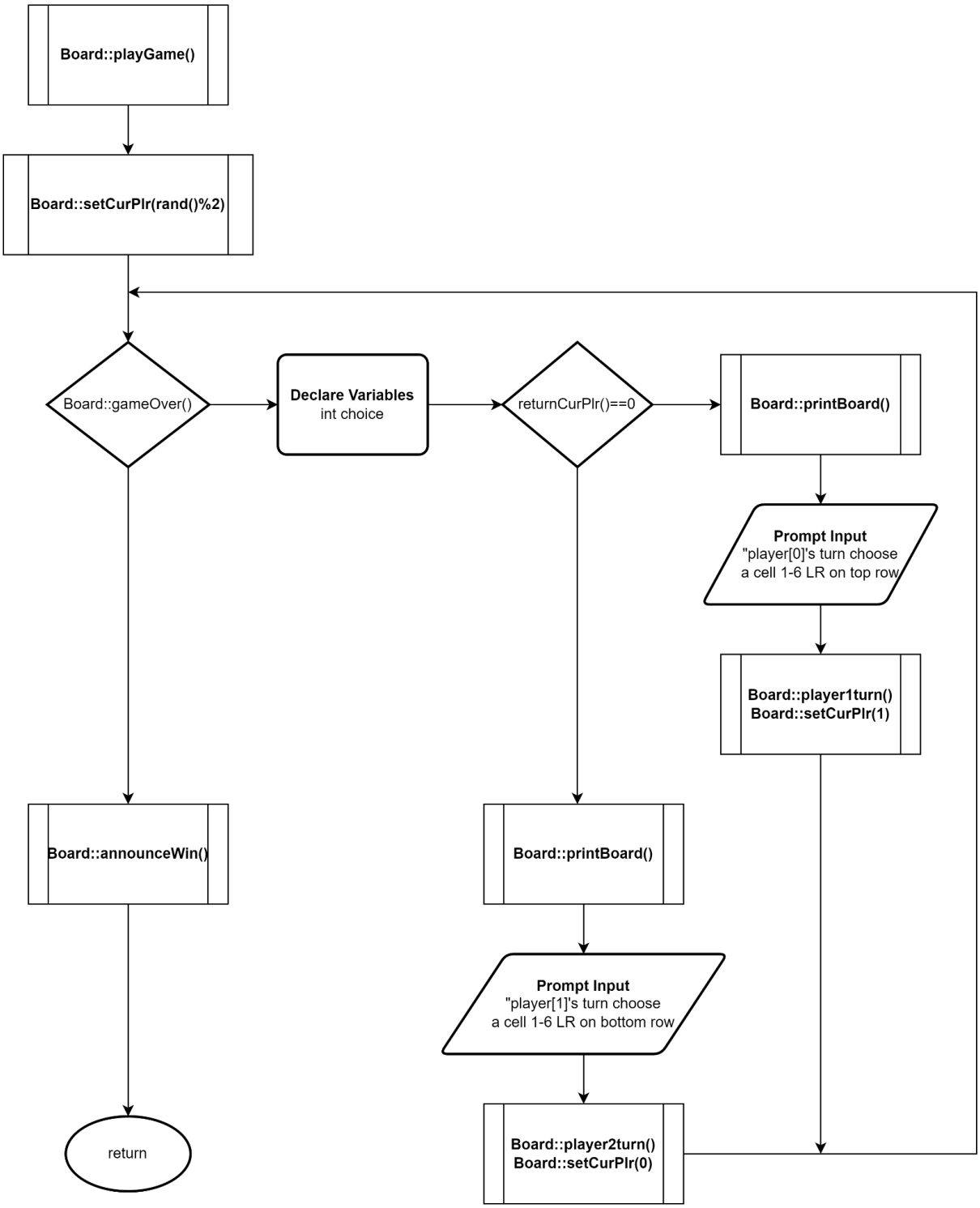
3.Algorithms (Choose at least 1 from each category)

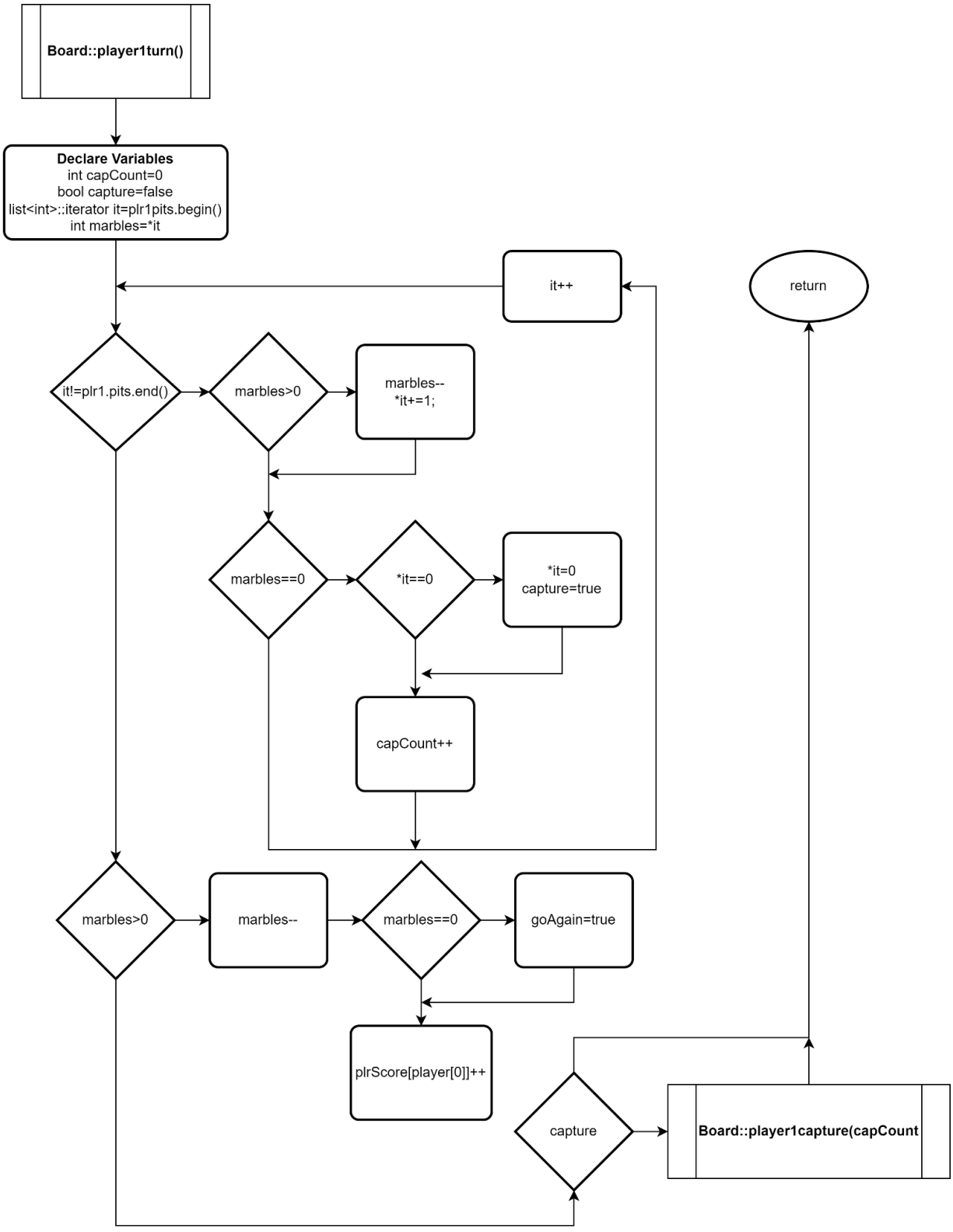
- 1.Non-mutating algorithms
  - 1.for_each
  - 2.find **Line 53 in "SimGame.h". Finds a random number for bot play**
  - 3.count
  - 4.equal
  - 5.search
- 2.Mutating algorithms
  - 1.copy **Line 50 in "SimGame.h". Copies the set of plays for bot to list**
  - 2.Swap
  - 3.Transform
  - 4.Replace
  - 5.fill
  - 6.Remove
  - 7.Random_Shuffle
- 3.Organization

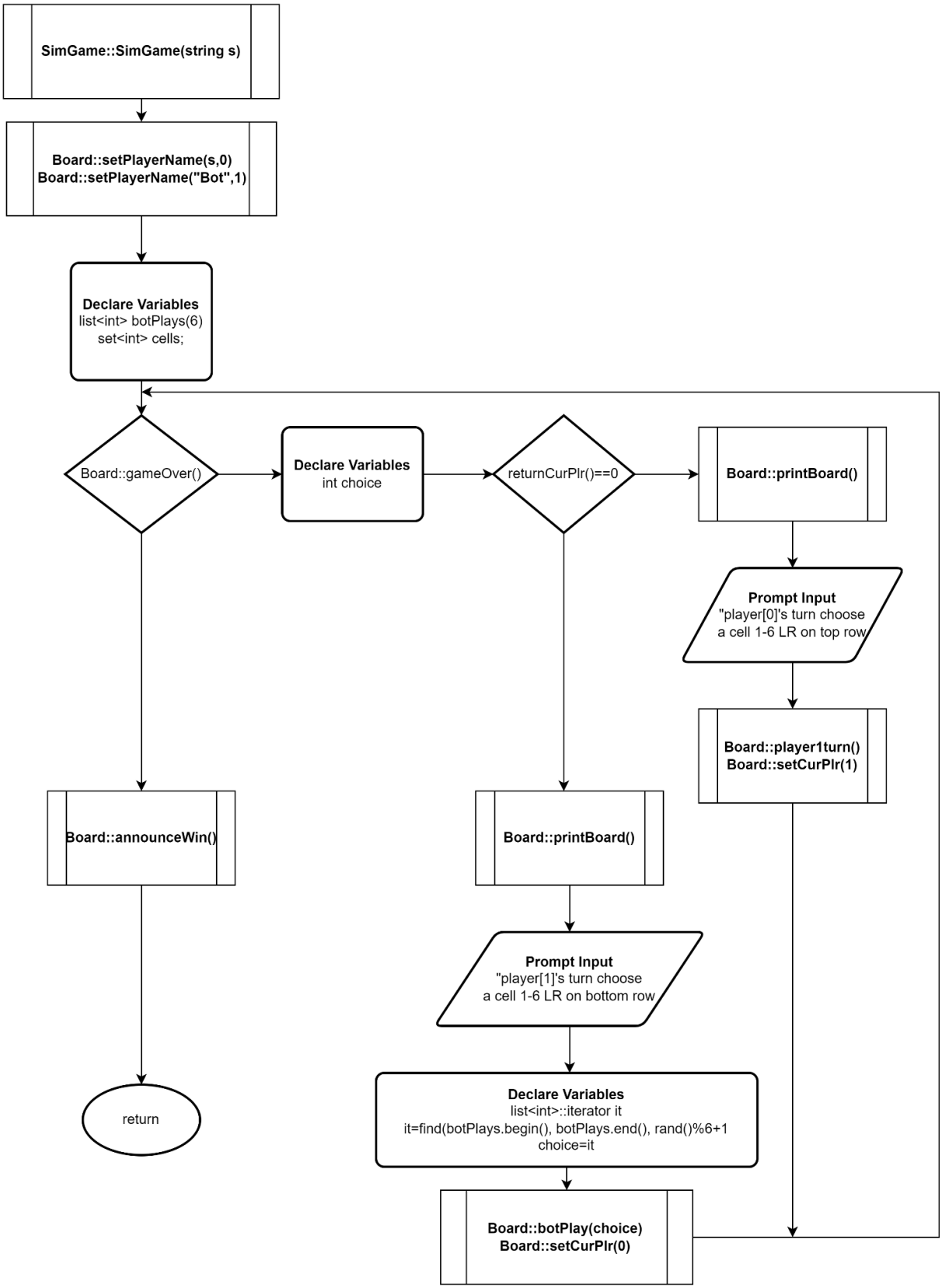
1.Sort
2.Binary search
3.merge
4.inplace_merge
5.Minimum and maximum **Line 798 in “Board.h”. Takes the max between the players’ scores to determine the winner.**

Flowchart


Author: Paul Ingram
Created On: May 2, 2022, 11PM
Purpose: Mancala v5
System Libraries
cstdlib
string
iostream
time.h
algorithm
list
map
stack
queue
User Libraries
Board.h
SimGame.h
Global Constants
none
Function Prototypes
none
main
Set Random Number Seed
and
Declare Variables
srand(time(NULL))
int gameMode
Prompt Input
"Main Menu..."
"Enter 1) 1v1"
"Enter 2) 1vBot"
gameMode==1
Declare Variables
string player[2]
int curPlr=rand()%2
Prompt Input
"Enter Player 1 Name: "
"Enter Player 2 Name: "
Board Constructor
board(player[0], player[1])
Board::playGame()


Board::playGame()
Board::setCurPlr(rand()%2)
Board::gameOver()
Declare Variables
int choice
returnCurPlr()==0
Board::printBoard()
Prompt Input
"player[0]'s turn choose
a cell 1-6 LR on top row
Board::player1turn()
Board::setCurPlr(1)
Board::announceWin()
Board::printBoard()
Prompt Input
"player[1]'s turn choose
a cell 1-6 LR on bottom row
return
Board::player2turn()
Board::setCurPlr(0)


Board::player1turn()
Declare Variables
int capCount=0
bool capture=false
list<int>::iterator it=plr1pits.begin()
int marbles=*it
it++
return
it!=plr1.pits.end()
marbles>0
marbles--
*it+=1;
marbles==0
*it==0
*it=0
capture=true
capCount++
marbles>0
marbles--
marbles==0
goAgain=true
plrScore[player[0]]++
capture
Board::player1capture(capCount


SimGame::SimGame(string s)
Board::setPlayerName(s,0)
Board::setPlayerName("Bot",1)
Declare Variables
list<int> botPlays(6)
set<int> cells;
Board::gameOver()
Declare Variables
int choice
returnCurPlr()==0
Board::printBoard()
Prompt Input
"player[0]'s turn choose
a cell 1-6 LR on top row
Board::player1turn()
Board::setCurPlr(1)
Board::announceWin()
Board::printBoard()
Prompt Input
"player[1]'s turn choose
a cell 1-6 LR on bottom row
return
Declare Variables
list<int>::iterator it
it=find(botPlays.begin(), botPlays.end(), rand()%6+1
choice=it
Board::botPlay(choice)
Board::setCurPlr(0)

Pseudocode

Get game mode from user
If user selects 1v1 get both user names and construct a Board class
Set the current player to a random number between 0 and 1
While the game is not over
    If the current player is equal to 0 gather play from the user
    Update the game history
    Update the game board with the user's choice while checking for second chances and captures
    Set the current player to 1
    If the current player is equal to 1 gather play from the user
    Update the game history
    Update the game board with the user's choice while checking for second chances and captures
    Set the current player to 1
Once the game is over announce the winner and their score

Else gather user name and construct a SimGame class
Do the same as if it was a 1v1 but generate a random number for the bot player

UML Diagram