COMS 4996 Parallel Functional Programming Final Project – Fall 2021 - Bingo

Deepakraj Dharmapuri Selvakumar - dd3068

December 20, 2021

1 Description

Bingo is a game where each player matches numbers called out by the game host in their respective bingo board. A Bingo board is a 5 x 5 matrix where each cell has randomly placed unique numbers from 1 to 25. When the game host calls out a number, each player strikes out the number from their board. When the player has a row or a column or any of the diagonals of strikes, then that player wins, if both player gets the strikes at the same time, then it's a draw.

2 Data

Since its a casual and fun game I wanted to make, I made it interactive where the user can give the number of players and dimensions as an input and as in real scenario, we will have "bingo" cards generated for the user. and then checks for bingo will happen in parallel. But in reality, 5x5 boards are small and very fast to calculate in a system. Hence I increased the dimensions to 25x25 and 50x50. I also increased the number of players from 2 to 5, 50, 100 to play with the complexity and see how much of an improvement the parallel algorithm can make.

3 Strategy

I expressed these as matrix and used pretty print matrix from package Data.Matrix to display Bingo Cards to the user. There were three points of focus where I could parallelize here.

- Parallelize strike box when the game host calls a number

- Parallelize checkWin - which basically checks row, col, diag in parallel.

- Parallelize the above check for all the players

In order to achieve the above, I used the concept of parList rseq for parallelizing the strikeBox and checkWin for individual players while I used parPair for parallelizing row check and column check.

4 Actual Bingo Game and Output

Listing 1: Bingo

```
(base) deepakrajds@DEEPAKRAJs-MBP bingo-hs % time .stack-work/
    install/x86_64-osx/
    bb33c3913c44d480d21e221088f62d837946cadc12824c4ae68ba3310462aca8
    /8.10.7/bin/bingo-hs-exe test/testcases/player1-5.txt test/
    testcases/player2-5.txt test/testcases/gh5.txt 5 +RTS -N8 -
    ls

      4   6   9  25  24
     20  19  17  13   7
     16   3   1  11  15
      2  12  14  23  21
     22  18   5   8  10


     22   1  25  15  21
     19   6  14   3  18
     23  10   4  13  20
      2  12  17   9   7
     11   5   8  24  16

Game Host calls 9!!
Player 1

       "4"    "6"  "'X'"   "25"   "24"
      "20"   "19"   "17"   "13"    "7"
      "16"    "3"    "1"   "11"   "15"
       "2"   "12"   "14"   "23"   "21"
      "22"   "18"    "5"    "8"   "10"

Player 2

      "22"    "1"   "25"   "15"   "21"
      "19"    "6"   "14"    "3"   "18"
      "23"   "10"    "4"   "13"   "20"
       "2"   "12"   "17"  "'X'"    "7"
      "11"    "5"    "8"   "24"   "16"

Game Host calls 20!!
Player 1

       "4"    "6"  "'X'"   "25"   "24"
     "'X'"   "19"   "17"   "13"    "7"
      "16"    "3"    "1"   "11"   "15"
       "2"   "12"   "14"   "23"   "21"
```

```
      "22"   "18"    "5"    "8"   "10"

Player 2

      "22"    "1"   "25"   "15"   "21"
      "19"    "6"   "14"    "3"   "18"
      "23"   "10"    "4"   "13"  "'X'"
       "2"   "12"   "17"  "'X'"    "7"
      "11"    "5"    "8"   "24"   "16"

Game Host calls 22!!
Player 1

       "4"    "6"  "'X'"   "25"   "24"
     "'X'"   "19"   "17"   "13"    "7"
      "16"    "3"    "1"   "11"   "15"
       "2"   "12"   "14"   "23"   "21"
     "'X'"   "18"    "5"    "8"   "10"

Player 2

     "'X'"    "1"   "25"   "15"   "21"
      "19"    "6"   "14"    "3"   "18"
      "23"   "10"    "4"   "13"  "'X'"
       "2"   "12"   "17"  "'X'"    "7"
      "11"    "5"    "8"   "24"   "16"

Game Host calls 11!!
Player 1

       "4"    "6"  "'X'"   "25"   "24"
     "'X'"   "19"   "17"   "13"    "7"
      "16"    "3"    "1"  "'X'"   "15"
       "2"   "12"   "14"   "23"   "21"
     "'X'"   "18"    "5"    "8"   "10"

Player 2

     "'X'"    "1"   "25"   "15"   "21"
      "19"    "6"   "14"    "3"   "18"
      "23"   "10"    "4"   "13"  "'X'"
       "2"   "12"   "17"  "'X'"    "7"
     "'X'"    "5"    "8"   "24"   "16"

Game Host calls 16!!
Player 1

       "4"    "6"  "'X'"   "25"   "24"
```

```
    "'X'"   "19"   "17"   "13"    "7"
    "'X'"    "3"    "1"  "'X'"   "15"
      "2"   "12"   "14"   "23"   "21"
    "'X'"   "18"    "5"    "8"   "10"

Player 2

    "'X'"    "1"   "25"   "15"   "21"
     "19"    "6"   "14"    "3"   "18"
     "23"   "10"    "4"   "13"  "'X'"
      "2"   "12"   "17"  "'X'"    "7"
    "'X'"    "5"    "8"   "24"  "'X'"

Game Host calls 19!!
Player 1

      "4"    "6"  "'X'"   "25"   "24"
    "'X'"  "'X'"   "17"   "13"    "7"
    "'X'"    "3"    "1"  "'X'"   "15"
      "2"   "12"   "14"   "23"   "21"
    "'X'"   "18"    "5"    "8"   "10"

Player 2

    "'X'"    "1"   "25"   "15"   "21"
    "'X'"    "6"   "14"    "3"   "18"
     "23"   "10"    "4"   "13"  "'X'"
      "2"   "12"   "17"  "'X'"    "7"
    "'X'"    "5"    "8"   "24"  "'X'"

Game Host calls 7!!
Player 1

      "4"    "6"  "'X'"   "25"   "24"
    "'X'"  "'X'"   "17"   "13"  "'X'"
    "'X'"    "3"    "1"  "'X'"   "15"
      "2"   "12"   "14"   "23"   "21"
    "'X'"   "18"    "5"    "8"   "10"

Player 2

    "'X'"    "1"   "25"   "15"   "21"
    "'X'"    "6"   "14"    "3"   "18"
     "23"   "10"    "4"   "13"  "'X'"
      "2"   "12"   "17"  "'X'"  "'X'"
    "'X'"    "5"    "8"   "24"  "'X'"

Game Host calls 14!!
```

```
Player 1

       "4"    "6"  "'X'"   "25"   "24"
    "'X'"  "'X'"   "17"   "13"  "'X'"
    "'X'"    "3"    "1"  "'X'"   "15"
       "2"   "12"  "'X'"   "23"   "21"
    "'X'"   "18"    "5"    "8"   "10"

Player 2

    "'X'"    "1"   "25"   "15"   "21"
    "'X'"    "6"  "'X'"    "3"   "18"
     "23"   "10"    "4"   "13"  "'X'"
       "2"   "12"   "17"  "'X'"  "'X'"
    "'X'"    "5"    "8"   "24"  "'X'"

Game Host calls 10!!
Player 1

       "4"    "6"  "'X'"   "25"   "24"
    "'X'"  "'X'"   "17"   "13"  "'X'"
    "'X'"    "3"    "1"  "'X'"   "15"
       "2"   "12"  "'X'"   "23"   "21"
    "'X'"   "18"    "5"    "8"  "'X'"

Player 2

    "'X'"    "1"   "25"   "15"   "21"
    "'X'"    "6"  "'X'"    "3"   "18"
     "23"  "'X'"    "4"   "13"  "'X'"
       "2"   "12"   "17"  "'X'"  "'X'"
    "'X'"    "5"    "8"   "24"  "'X'"

Game Host calls 8!!
Player 1

       "4"    "6"  "'X'"   "25"   "24"
    "'X'"  "'X'"   "17"   "13"  "'X'"
    "'X'"    "3"    "1"  "'X'"   "15"
       "2"   "12"  "'X'"   "23"   "21"
    "'X'"   "18"    "5"  "'X'"  "'X'"

Player 2

    "'X'"    "1"   "25"   "15"   "21"
    "'X'"    "6"  "'X'"    "3"   "18"
     "23"  "'X'"    "4"   "13"  "'X'"
       "2"   "12"   "17"  "'X'"  "'X'"
```

```
    "'X'"    "5"  "'X'"   "24"  "'X'"

Game Host calls 23!!
Player 1

      "4"    "6"  "'X'"   "25"   "24"
    "'X'"  "'X'"   "17"   "13"  "'X'"
    "'X'"    "3"    "1"  "'X'"   "15"
      "2"   "12"  "'X'"  "'X'"   "21"
    "'X'"   "18"    "5"  "'X'"  "'X'"

Player 2

    "'X'"    "1"   "25"   "15"   "21"
    "'X'"    "6"  "'X'"    "3"   "18"
    "'X'"  "'X'"    "4"   "13"  "'X'"
      "2"   "12"   "17"  "'X'"  "'X'"
    "'X'"    "5"  "'X'"   "24"  "'X'"

Game Host calls 3!!
Player 1

      "4"    "6"  "'X'"   "25"   "24"
    "'X'"  "'X'"   "17"   "13"  "'X'"
    "'X'"  "'X'"    "1"  "'X'"   "15"
      "2"   "12"  "'X'"  "'X'"   "21"
    "'X'"   "18"    "5"  "'X'"  "'X'"

Player 2

    "'X'"    "1"   "25"   "15"   "21"
    "'X'"    "6"  "'X'"  "'X'"   "18"
    "'X'"  "'X'"    "4"   "13"  "'X'"
      "2"   "12"   "17"  "'X'"  "'X'"
    "'X'"    "5"  "'X'"   "24"  "'X'"

Game Host calls 13!!
Player 1

      "4"    "6"  "'X'"   "25"   "24"
    "'X'"  "'X'"   "17"  "'X'"  "'X'"
    "'X'"  "'X'"    "1"  "'X'"   "15"
      "2"   "12"  "'X'"  "'X'"   "21"
    "'X'"   "18"    "5"  "'X'"  "'X'"

Player 2

    "'X'"    "1"   "25"   "15"   "21"
```

```
    "'X'"    "6"  "'X'"  "'X'"   "18"
    "'X'"  "'X'"    "4"  "'X'"  "'X'"
      "2"   "12"   "17"  "'X'"  "'X'"
    "'X'"    "5"  "'X'"   "24"  "'X'"

Game Host calls 2!!
Player 1

      "4"    "6"  "'X'"   "25"   "24"
    "'X'"  "'X'"   "17"  "'X'"  "'X'"
    "'X'"  "'X'"    "1"  "'X'"   "15"
    "'X'"   "12"  "'X'"  "'X'"   "21"
    "'X'"   "18"    "5"  "'X'"  "'X'"

Player 2

    "'X'"    "1"   "25"   "15"   "21"
    "'X'"    "6"  "'X'"  "'X'"   "18"
    "'X'"  "'X'"    "4"  "'X'"  "'X'"
    "'X'"   "12"   "17"  "'X'"  "'X'"
    "'X'"    "5"  "'X'"   "24"  "'X'"

BINGO!!! *** Player 2 Won!! ***
 test/testcases/player1-5.txt test/testcases/player2-5.txt  5
   +RTS -N8 -ls  0.01s user 0.01s system 11% cpu 0.170 total
```

FOR THE PURPOSES OF TESTING AND INCREASING COMPLEXITY, WE WILL HIDE THE OUTPUT OF THE MATRIX HENCEFORTH

5 Tests and Inferences

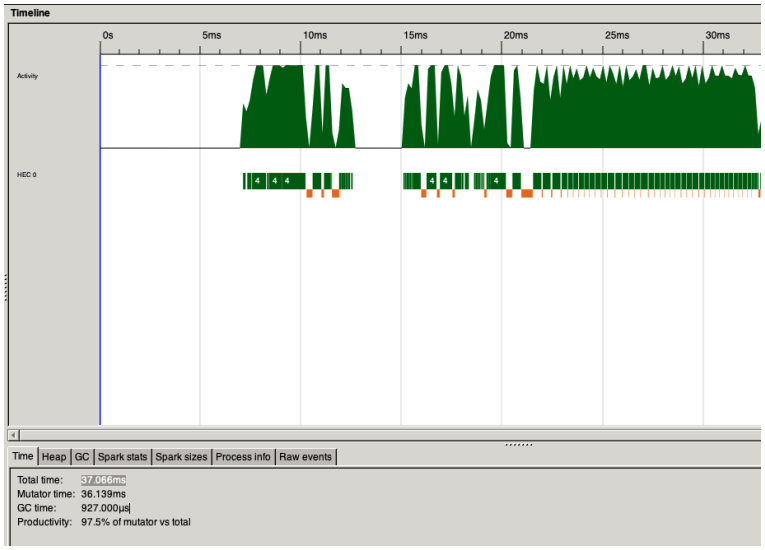
5.1 The Two Player Scenario

At first, I started of with only two players, comparing 25x25 boards and 50x50 boards. It rather seemed to have increased the time than decreasing when I parallelized which shows that in case of two players, the serialization algorithm was faster. Below are the screenshots and inferences.

25x25 Board

Serial:

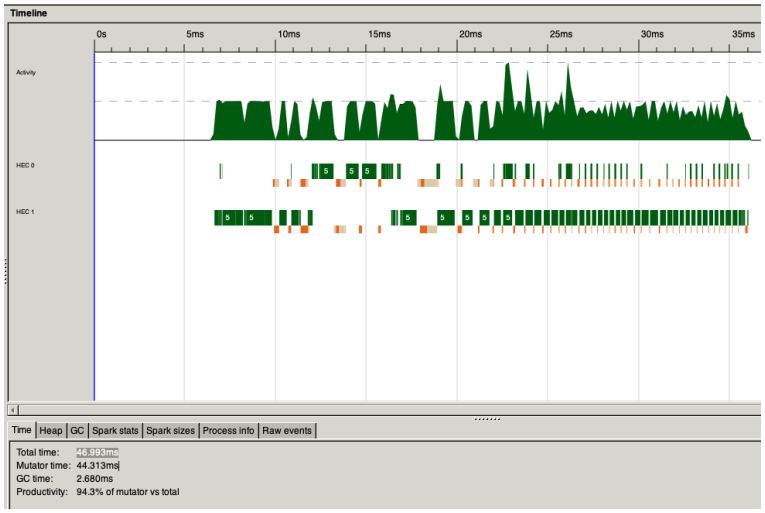
test/testcases/player1-25.txt test/testcases/player2-25.txt 25 +RTS -N1 -ls



Total Time: 37.066ms

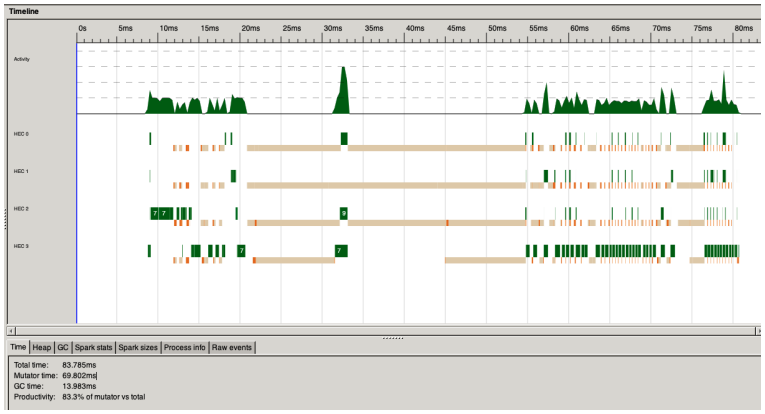
Parallel - 2 cores

test/testcases/player1-25.txt test/testcases/player2-25.txt 25 +RTS -N2 -ls



Parallel - 4 cores

test/testcases/player1-25.txt test/testcases/player2-25.txt 25 +RTS -N4 -ls



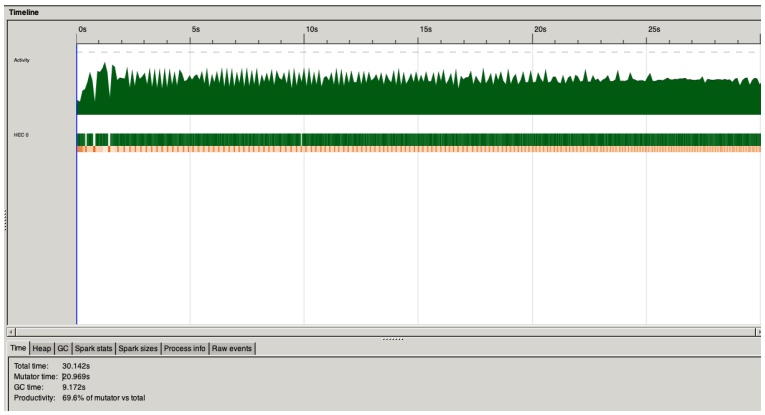
Inference

As we can see, the parallelization seemed to have increased as compared. It seemed to have wasted a lot of time in garbage collecting and we can see spikes in all of the cores at only some point. This clearly shows that parallelization doesnt help in have limited players and having these checks. This was the similar or has worse case for 50x50 board with two players. Clearly increasing the board size didnt help in parallelization and decrease in time. Why is this happening? Probably because, ‘parList rseq‘ doesnt really help here in case of two players as strikebox and checksolution has less things to parallelize and it ends up adding overhead to it. Total sparks generated in 2 cores is about 4k of which only 136 got converted while the rest being garbage collected while its similar in 4 cores about twice got coverted.

5.2 The 200 Players scenario with 50x50 board

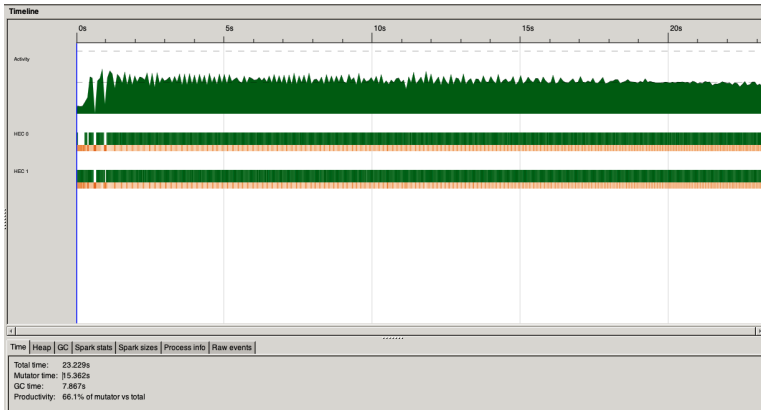
In this scenario, we are determined to make use of parList rseq for strikeBox and CheckWin so I have increased the number of players and increased the board dimensions to 50x50

Serial - 1 core ./test/testcases/players-50.txt ./test/testcases/gh-50.txt 50 +RTS -N1 -ls



Total Time: 30.142s

Parallel - 2 core ./test/testcases/players-50.txt ./test/testcases/gh-50.txt 50 +RTS -N2 -ls

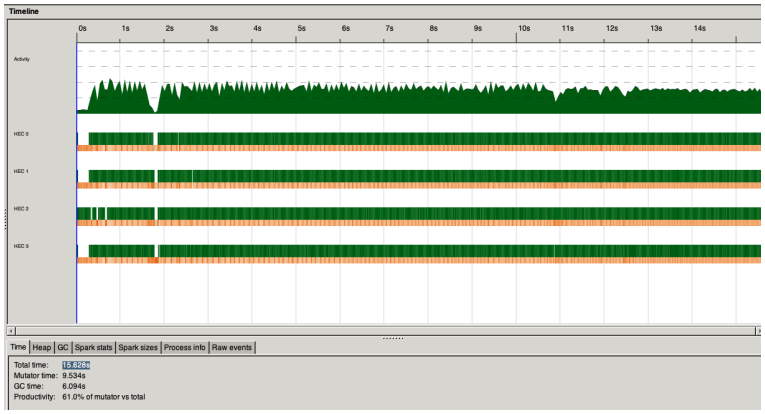


Total Time: 23.229s

Inference

We can clearly see an improvement here from the serial algorithm. There is a decrease in time by 23%. The ‘parList rseq‘ plays a crucial role in achieving this strat. Compared to the previous scenario, it doesnt spend too much time in gargabe collecting. Both cores have been fully utilized here as there are constant spikes ar regular intervals. Although there are instances of garbage collection but overall it contributes significantly less. In case of sparks being generated, there are about 1.5 millions sparks of which almost most of it are converted and it augments the above inference.

Parallel - 4 core ./test/testcases/players-50.txt ./test/testcases/gh-50.txt 50 +RTS -N4 -ls

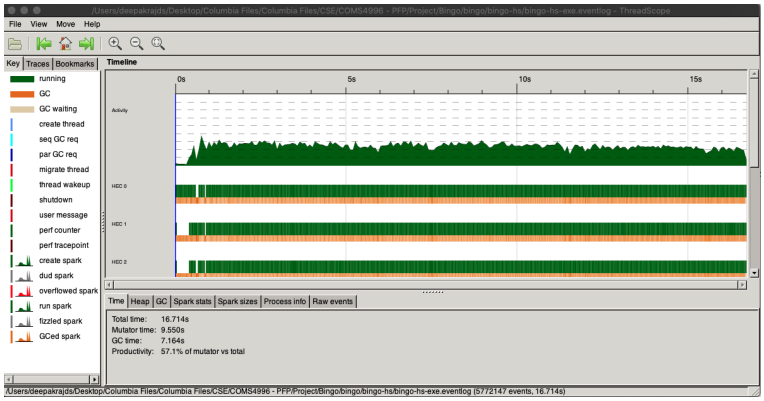


Total Time: 15.628s

Inference

We can clearly see an improvement here from the serial algorithm. There is a decrease in time by 50%. The ‘parList rseq‘ plays a crucial role in achieving this strat. Compared to the previous scenario, it doesnt spend too much time in gargabe collecting. All the four cores have been fully utilized here as there are constant spikes are regular intervals. Although there are instances of garbage collection but overall it contributes significantly less. In case of sparks being generated, there are about 1.5 millions sparks of which almost most of it are converted and it augments the above inference. In case of sparks being generated, there are about 1.5 millions sparks of which almost most of it are converted and it augments the above inference.

Parallel - 8 core ./test/testcases/players-50.txt ./test/testcases/gh-50.txt 50 +RTS -N4 -ls



Total Time: 16.714s

Inference

We do not see much of an improvement here when we used 8 cores, my machine has quad core but that doesnt necessarily mean we can only run 4 threads at a time, it all depends on the underlying OS / threading library on how to schedule these threads as we can see all the "eight" threads have been utilized similar to the four thread one. We see a slight improvement in the time as compared to the previous test with only four core. This is probably time taken between the scheduling on 8 threads in four cores or something along those lines.

6 Overall Inference

Summarizing all the inferences from above

- Parallelization doesnt seem to help much in case of only two players even if we increase the board dimensions mainly due to because of lot of time spent in garbage collecting as compared to the serial algorithm and ‘parList rseq‘ not being utilized much.
- In case of 200 players with a larger board, parallelization helps us tremendously as we can see constant spikes in all the cores with minimal time spent in garbage collecting. The use of ‘parList rseq‘ is clearly being shown here.

7 Appendix - Code

Please find the code in proper format in zip file. Latex has some trouble displaying these.

```
{-

Final Project - Bingo

Deepakraj Dharmapuri Selvakumar - dd3068

-}

{-# LANGUAGE GeneralizedNewtypeDeriving #-}
{-# LANGUAGE EmptyDataDecls #-}
{-# LANGUAGE TypeSynonymInstances #-}
{-# LANGUAGE FlexibleInstances #-}
{-# LANGUAGE FlexibleContexts #-}
{-# LANGUAGE UndecidableInstances #-}

import qualified Data.Map.Strict as Map
import qualified Data.List as List
import Data.List.Split as Split
import qualified Data.Matrix as Mat
import System.Random (newStdGen, RandomGen, mkStdGen)
import System.Random.Shuffle (shuffle')
import Control.Monad(forM)
import System.Exit(die)
import Control.Parallel.Strategies hiding (parPair)
import Control.Monad.Parallel as Mp
import System.IO.Error
import System.Environment(getArgs, getProgName)

type MatBox = (Int, Int, String) -- x, y, val
type Pattern = (Int, Int) -- row, col, diag pattern

instance {-# OVERLAPPING #-} Show MatBox => Show (Int, Int,
    String)
  where
    show (x, y, val) = (show val)

parPair :: Strategy (a, b)
parPair (a, b) = do
  a' <- rpar a
  b' <- rpar b
  return (a', b')

-- Gen List for dimensions --
genList :: Int -> [Int]
genList x = [1..x]
```

```
-- In case players without predefined bingo boards, the
   players get assigned random bingo cards --
getShuffles :: RandomGen gen => [Int] -> Int -> gen -> [[Int]]
getShuffles mat player gen = ((Split.chunksOf 5 cards) !! (
   player `mod` 5 ) )
                            where
                              cards = map (\cur -> shuffle' cur
                                 (length cur) gen) spurs
                              spurs = take 50 (map (\cr ->
                                 shuffle' cr (length cr) ngn)
                                 permt)
                              permt = take 1000 (List.
                                 permutations mat)
                              ngn = mkStdGen 40

getMatrix :: [Int] -> Mat.Matrix Int
getMatrix a = Mat.fromList x x a where x = round ( sqrt (
   fromIntegral $ length a) )

-- Get Bingo Board for a player --
getBoard :: Int -> Mat.Matrix Int -> Mat.Matrix MatBox
getBoard dim matc = Mat.matrix dim dim $ \( i, j ) -> ( i, j,
   show (Mat.getElem i j matc)  )

-- For striking, we anyways have to traverse the whole matrix,
    since we go row by row --
searchRow :: (Int, Int) -> Mat.Matrix MatBox -> Int -> (Bool,
   Int)
searchRow (x, y) card val | y > (Mat.ncols card) = (False, y)
                          | vv == (show val) = (True, y)
                          | otherwise = searchRow (x, y + 1)
                             card val
                           where (rr, cc, vv) = (Mat.getElem x
                              y card)

searchMatrix :: Mat.Matrix MatBox -> (Int, Int) -> Int -> (Mat
   .Matrix MatBox, (Int, Int))
searchMatrix card (x, y) val | state = ( (Mat.setElem (x, col,
    (show 'X')) (x, col) card), (x, col) )
                             | otherwise = searchMatrix card (
                                x + 1, y) val
                               where (state, col) = searchRow
                                  (x, y) card val

-- Row, Col, Diag checks --
checkRow :: (Int, Int) -> Mat.Matrix MatBox -> Bool
checkRow (x, y) card | y > (Mat.ncols card) = True
```

```
                        | vv /= (show 'X') = False
                        | otherwise = checkRow (x, y + 1) card
                        where (rr, cc, vv) = (Mat.getElem x y
                            card)

checkCol :: (Int, Int) -> Mat.Matrix MatBox -> Bool
checkCol (x, y) card | x > (Mat.nrows card) = True
                        | vv /= (show 'X') = False
                        | otherwise = checkCol (x + 1, y) card
                        where (rr, cc, vv) = (Mat.getElem x y
                            card)

checkDiag :: (Int, Int) -> Mat.Matrix MatBox -> Bool
checkDiag (x, y) card | x > (Mat.nrows card) = True
                        | vv /= (show 'X') = False
                        | otherwise = checkDiag ( x+1, y+1) card
                          where (rr, cc, vv) = (Mat.getElem x y
                              card)

-- Check Sol or checkWin --
checkSolution :: (Int, Int) -> Mat.Matrix MatBox -> Bool
checkSolution (x, y) card | x == y = cr || cc || (checkDiag
    (1, 1) card)
                          | otherwise = cr || cc
                          where (cr, cc) = ((checkRow (x, 1)
                              card), (checkCol (1, y) card)) `
                              using` parPair -- parallelized

strikeBox :: Int -> Mat.Matrix MatBox -> (Mat.Matrix MatBox, (
    Int, Int))
strikeBox val card = searchMatrix card (1, 1) val

printMatrix :: (Int, Mat.Matrix MatBox) -> String
printMatrix (player, mat) = "Player " ++ (show player) ++ "\n"
     ++ (Mat.prettyMatrix mat)

-- playGame for each number the game host calls --
playGame :: [Int] -> [Mat.Matrix MatBox] -> IO ()
playGame [] _ = putStrLn "Game Over"
playGame (x:xs) pcards = do
  let rest = map (\card -> strikeBox x card) pcards `using`
      parList rseq -- parallelized
  let ncards = map (\(mat, (x, y)) -> mat ) rest
  let sols = map (\(mat, (x, y)) -> checkSolution (x, y) mat )
      rest `using` parList rseq -- parallelized
  let winners = filter (\(player, state) -> state) $ zip [1..(
      (length pcards) ::Int )] sols
  -- putStrLn $ "Game Host calls " ++ (show x) ++ "!!"
```

```
  -- Prelude.mapM_ (putStrLn . printMatrix) (zip [1..( (length
      pcards) ::Int )] ncards) -- uncomment for output
  if null winners then
    playGame xs ncards
  else
    if (length winners) == 1 then
      die $ "BINGO!!! *** Player " ++ (show ( fst (head
         winners) ) ) ++ " Won!! ***"
    else
      die $  "Its a Draw!!"

--- just added for test cases ---
readLines :: FilePath -> IO [String]
readLines = fmap lines . readFile

makeInteger :: [String] -> [Int]
makeInteger = map read
----         *******        -----

main :: IO ()
main = do args <- getArgs
          case args of
            [d, p] -> do
              let dimensions = read d :: Int
              let players = read p :: Int
              rng <- newStdGen
              let mcards = map (\player -> (getShuffles (
                 genList (dimensions * dimensions) ) player (
                 mkStdGen player)) !! ( player `mod` 5 ) )
                 [1..(players :: Int)]
              let ghcard = shuffle' (genList (dimensions *
                 dimensions)) (dimensions * dimensions) rng
              putStrLn "These are the current players assigned
                  bingo cards"
              Prelude.mapM_ (putStrLn . Mat.prettyMatrix .
                 getMatrix) mcards
              let boards = map (\x -> getBoard dimensions (
                 getMatrix x)) mcards
              playGame ghcard boards
            [pl1f, pl2f, ghf, d] -> do
              p1b <- readLines pl1f
              p2b <- readLines pl2f
              ghb <- readLines ghf
              let pl1m = makeInteger p1b
              let pl2m = makeInteger p2b
              let ghm = makeInteger ghb
              let players = 2
              let dimensions = read d :: Int
```

```
            let mcards = [pl1m, pl2m]
            let ghcard = ghm
            Prelude.mapM_ (putStrLn . Mat.prettyMatrix .
               getMatrix) mcards
            let boards = map (\x -> getBoard dimensions (
               getMatrix x)) mcards
            playGame ghcard boards
          [players, ghf, d] -> do
            scards <- readLines players
            let hycards = map words scards
            let dimensions = read d :: Int
            let mcards = map makeInteger hycards
            let players = (length mcards) ::Int
            ghb <- readLines ghf
            let ghm = makeInteger ghb
            let ghcard = ghm
            putStrLn "These are the current players assigned
                bingo cards"
            -- Prelude.mapM_ (putStrLn . Mat.prettyMatrix .
               getMatrix) mcards -- uncomment for initial
               card outputs
            let boards = map (\x -> getBoard dimensions (
               getMatrix x)) mcards
            putStrLn "Start game"
            playGame ghcard boards
          _ -> do pn <- getProgName
                  die $ "Usage: " ++ pn ++ " <
                     board_dimensions> <total_players> or <
                     player1_file> <player2_file> <
                     gamehost_file> <dimensions> or <
                     players_file> <gamehost_file> <
                     dimensions>"
`catchIOError` \ e -> die $ case ioeGetFileName e of
       Just fn | isDoesNotExistError e -> fn ++ ": No such
          file"
               | isPermissionError e -> fn ++ ": Permission
                  denied"
       _       | isUserError e -> "Usage: ./bingo <
          board_dimensions> <total_players>"
               | otherwise -> show e
```