Requirements:

* Shall initialize grid plane and at least 2 players – 10%
* Shall display grid lines/boundaries, player light bikes, light walls – 10%
* Shall use keyboard inputs to control player bikes and program – 20%
  + WASD and arrow keys for player control – 15%
  + Escape to close the game – 5%
  + May use backspace to exit to main menu
* Shall use an algorithm for bot players input – 20%
* Shall check collisions – 20%
  + Check player coord against grid array – 10%
  + Delete color that died – 10%
* Shall refresh screen with updated information – 10%
* Shall display a win/lose/tie message – 10%
* May have a main menu to handle options and other features
  + Help menu
  + Set names menu
* May prompt for game settings
  + # of human players (max of 2) player 1 uses WASD; player 2 uses arrow keys
  + # of bot players (max of 2 for multiplayer and 3 for single player)
* May override quitting after game done to allow replayability
* May display status of players (name, alive/dead, direction)

External libraries included: curses and windows

Functional decomposition:

* main (uses struct and pointer for player information)
  + initProgram
    - initColors
  + menuHandler
    - displayMenu
    - menuInput
      * showHelp
      * showNames
        + loadNames (I/O)
        + saveNames (I/O)
  + initGame
    - loadNames
  + playerInput (dynamic memory and processing of array data)
  + computerInput
  + updateMoves
  + checkCollisions
    - killColors
  + updateScreen (graphics with ncurses)
  + checkWinner
  + endgame
  + clearMem

Detailed decomposition

main:

* Initializes program
* Loops into the menuHandler
* Continues the loop into the game once menu is done
* Initializes game grid and players and the rest of the game with initGame
* Loops into the game runtime which continues until
  + Checks playerInput returns a bool; if true, breaks the game runtime loop
  + Calls computerInput for each non-human player the settings provided
  + Calls updateMoves to adjust player movements
  + Calls checkCollisions
  + Calls updateScreen
  + Assigns a variable called “win” to checkWinner returns an int to determine who won
* If quiting, call endwin to tell ncurses to close

initGame:

* Sets srand to current time
* Creates a default player names file if one doesn’t exit
* Initializes ncurses settings
* Calls initColors

initColors:

* Sets colors blue, red, green, yellow, and grey
* Initializes color pairs with a number ID

menuHandler:

* Initialize menu selection options (start, help, game settings, set names, exit) and menu screen
* While menu doesn’t call to start the game
  + Call displayMenu
  + Call menuInput

displayMenu:

* Draw each menu item along with any relevant settings if it’s a game setting
* Highlights current menu option selection

menuInput:

* Check for W/S/up/down inputs to navigate menu
* Check for A/D/left/right inputs to change settings if on game setting
* Check for space on start, help, set names, and exit
  + Help calls showHelp
  + Set names calls showNames

showHelp:

* Displays help text
* Check for key press to exit help screen

showNames:

* Initializes array for names
* Calls loadNames which puts names into the names array
* Loops into infinite loop
  + Displays all four players and each of their names
  + Highlights current name to be modified or the exit at the bottom
  + Gets input from users
    - Up/down keys – move selection up and down respectively
    - Escape – closes program
    - Space
      * Calls saveNames and exits to main menu if on exit option
      * Puts a space in the name of the selected player
    - All characers within ASCII codes 32 and 126 inclusive are added to the name of the selected player
    - Backspace removes last character of the selected players name

loadNames:

* Opens file that stores player names
* Scans each line into the names array parameter
* Closes the file

saveNames:

* Opens file that stores player names
* Prints a line in the file with a player name for each name in the names array parameter
* Closes the file

initGame:

* Initialize the arena and status windows
* Set up gameArena and players

playerInput:

* Loop to grab all key inputs within the frame
* If one player
  + Check for WASD or arrow keys for respective movement
  + Adjust direction variable for player 1
* If two players
  + Check for WASD for respective movement for player 1
  + Adjust direction variable for player 1
  + Check for arrow keys for respective movement
  + Adjust direction variable for player 2
* If escape is hit, start closing game
* If backspace is hit, exit to main menu

computerInput:

* Determine what is in front of current bot player
* Don’t change direction if nothing is there
* Change directions 50%/50% if there is something in front

updateMoves:

* For each player:
  + Move in the direction of their status (^, v, <, >)
  + Update gameArena with location data of new player position and lightwalls

checkCollisions:

* Loop for each player:
  + If player status is dead, continue into loop
  + Check gameArena for any objects the player ran into player is dead if true
* Loop for each player:
  + If the player is dead, call killColor for their index number

killColor:

* Nested loop for y and x cords:
  + Grab char at coord
  + If char is color of dead player, replace with space and update gameArena

updateScreen:

* Loop for each player alive:
  + Draw their new position and light wall behind them with their respective color and direction
* Loop for each player:
  + Updating the status at the bottom of the screen (name, alive, direction)

checkWinner:

* Count how many players are alive
* If all players dead, return 0 to indicate a tie
* If one player is alive, return which player number 1-4 to indicate winner
* Otherwise, return -1 to indicate incomplete game

endGame:

* Clear status screen
* Display message with tie or name of player who won
* Indicate instructions to continue playing or exit the program
* Grab input: if escape, start closing game; if space, continue to menu

clearMem:

* Properly free memory from gameArena and player
* Clear all screens main, arena, status
* Refresh all screens
* Free memory from arena and status windows

Doc statement: no help received