Project 3

Due May 13, 2019 at 11:59 PM

This project specification is subject to change at any time for clarification. For this project you will be working with a partner. You are given the fix a given project that was developed by another developer. The project was a mancala game, the rules will be defined below. The existing game will work, but it is extremely buggy. The developer would compile the project using manual calls to g++ instead of using a Makefile and didn't use test driven design. The developer also did not use revision control, so you will be adding revision control with git. You are tasked with creating a Makefile for the project that will run compile/link a test program and the project. You will develop a set of tests have check correctness for the game. Once you have created the tests, you will then fix the existing code. You have been provided with the source files, and a skeleton test file.

You will find https://www.tutorialspoint.com/makefile/ helpful in developing your Makefile. The requirements for you Makefile are:

- Only used Conventional Macros should be declared (e.g. CXX, but not AR)
- The tests must be successfully run prior to building of mancala game
- For each target all true dependencies must be listed
- The Makefile must have a clean to remove object files, core dumps, etc.
- A conventional naming scheme must be used to organize the project files (source files, object files, etc.)
- The Makefile must build and run the tests and mancala game on the CSIF without modification

You need to create a git repository for your project. You should initialize it on github and then after cloning add the project files using the directory structure you have chosen. The following video https://www.youtube.com/watch?v=HVsySz-h9r4 has a very helpful tutorial on using git command line tools. You should commit progress as you make it and use branches so that you and your partner can develop in concurrently. You will include your .git directory in your submission. If you have not already gotten a github student developer pack, you should follow the directions here https://help.github.com/en/articles/applying-for-a-student-developer-pack. This will allow you to develop using a private repositories.

Mancala is a game that dates back well over 1,000 years and has many variations of rules. The following rules and board setup will be assumed for this project:

- The board has two player stores (these where their stones will be stored)
- The board has five pits on each side, each player has a set of five they can select from when it is their turn
- The initial board setup has four stones in each pit (see Figure 1)
- During a player's turn they select one of their five pits with stones in it and by moving counter clock wise place one stone in each pit.
    - If they reach their store, a stone is placed in their store
    - If they reach their opponents store, the store is skipped

- If the last stone is placed in the players store, the player gets another turn
- If the last stone lands in an empty pit of the player, the stone and all of the stones in the opponent's adjacent pit are stolen and put into the player's store
- If a player does not have any stones it their pits, they lose a turn
- When all stones are placed in the stores the game is over, the player with the most stones in their store wins

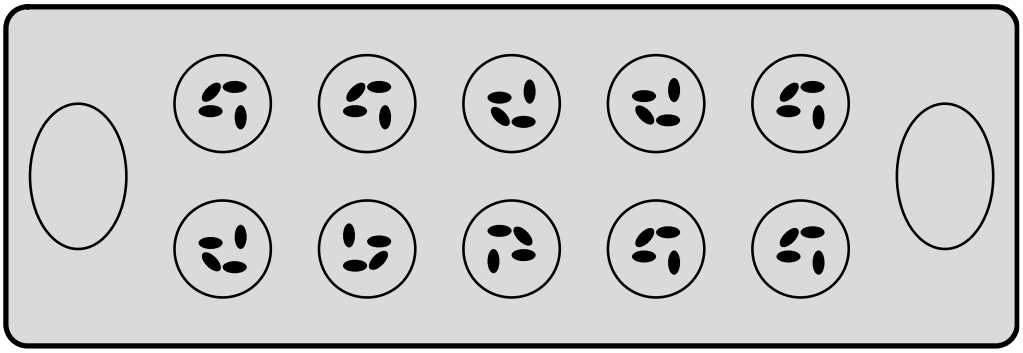

Figure 1. Initial Board Setup

An example of a normal turn is shown starting with the selected pit displayed in Figure 2. The result of the turn is shown in Figure 3.

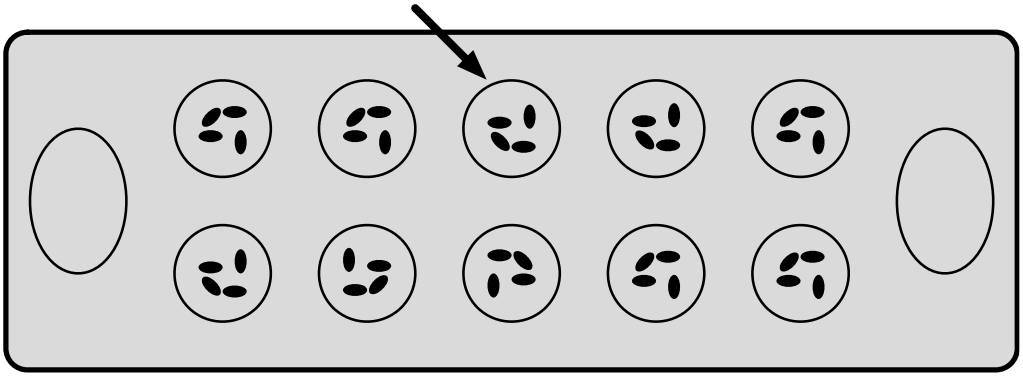

Figure 2. Pit Chosen for Turn

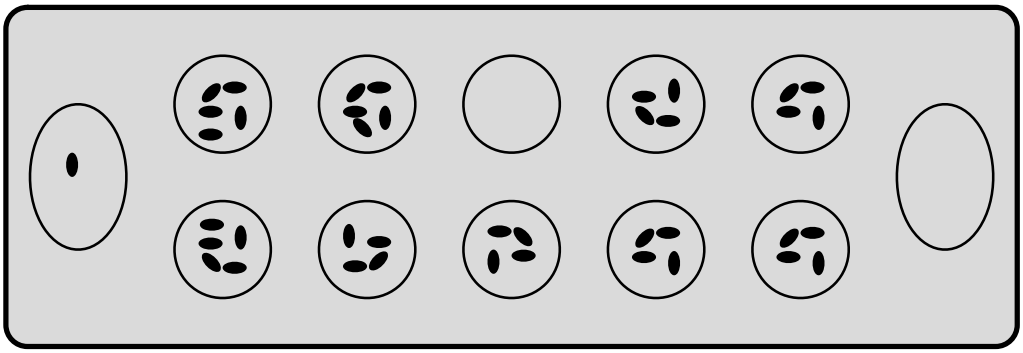

Figure 3. Result of Turn

The mancala game is implemented almost entirely in the `CMancalaBoard` class. The has `CMancalaBoard` the following member functions:

```
// Default constructor should initialize the board
CMancalaBoard();

// Constructor to specify a board setup with the amount in each
// pit, store and the turn
CMancalaBoard(int turn, const int pits[MANCALA_TOTAL_PITS],
const int stores[MANCALA_PLAYERS]);
// Resets the board back to initial setup
void ResetBoard();

// Returns which player's turn it is
int PlayerTurn() const;

// Returns the score of the player, -1 should be returned on bad
// parameter
int PlayerScore(int player) const;

// Returns the number of stones in the players pit, -1 should be
// returned on bad parameters
int PitStoneCount(int player, int pit);

// Returns true if the game is over
bool GameOver() const;

// Returns a string representation of the board of the form:
// P1          PITS
//       5   4   3   2   1
// /--------------------------\
// |   | 4 | 4 | 4 | 4 | 4 |   |
// | 0 |-------------------| 0 |
// |   | 4 | 4 | 4 | 4 | 4 |   |
// \--------------------------/
//       1   2   3   4   5
//              PITS          P2
std::string ToString() const;
operator std::string() const;

// Does a move for the player, by taking the stones from the
// specified pit. If the move failed, or it is not the player's
// turn Move should return false. If the move succeeds true
// should be returned
bool Move(int player, int pit);
```

You can unzip the given tgz file with utilities on your local machine, or if you upload the file to the CSIF, you can unzip it with the command:

```
tar -xzvf proj3given.tgz
```

You **must** submit the source file(s), the Makefile, .git fiels, and README.txt file, in a tgz archive. Do a `make clean` prior to zipping up your files so the size will be smaller. You can tar gzip a directory with the command:

```
tar -zcvf archive-name.tgz directory-name
```

Provide your interactive grading timeslot csv file in the tgz. The directions for filling it out have been posted.

You should avoid using existing source code as a primer that is currently available on the Internet. You **must** specify in your readme file any sources of code that you have viewed to help you complete this project. All class projects will be submitted to MOSS to determine if students have excessively collaborated. Excessive collaboration, or failure to list external code sources will result in the matter being referred to Student Judicial Affairs.

Helpful Hints

- Read through the guides that are provided on Canvas
- See http://www.cplusplus.com/reference/, it is a good reference for C++ built in functions and classes
- You may find the following line helpful for debugging your code:
  `std::cout<<"In "<<__FILE__<<" @ "<<__LINE__<<std::endl;`
  It will output the line string `"In F @ line: X"` where F is the name of the file and X is the line number the code is on. You can copy and paste it in multiple places and it will output that particular line number when it is on it.
- Make sure to use a tab and not spaces with the Makefile commands for the target
- Use a `.gitignore` file to ignore your object files, and output binaries
- Do not wait to the end to merge with you partner. You should merge your work together on a somewhat regular basis.