**Assignment 2, CSC384, Fall 2014**

Out: November 13th

Due: November 26th, 10pm (Electronic Submission, CDF)

This assignment is worth 14% of your final grade

**Be sure to include your name and student number as a comment in all submitted documents.**

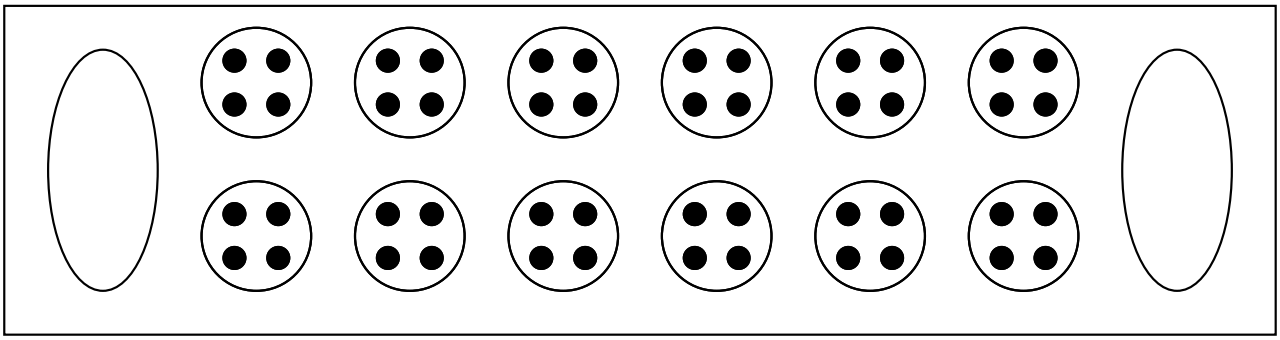
# 1 Introduction

In this assignment you will complete the problem description and a heuristic for a two player game called Mancala.

## 1.1 Game Rules

Mancala refers to a family of games played throughout the world. The playing space consists of two rows of pits, each row belonging to one player. In the version we deal with there is also a store, or score area, associated with each player. This score area is sometimes referred to as the mancala.

**North Player**



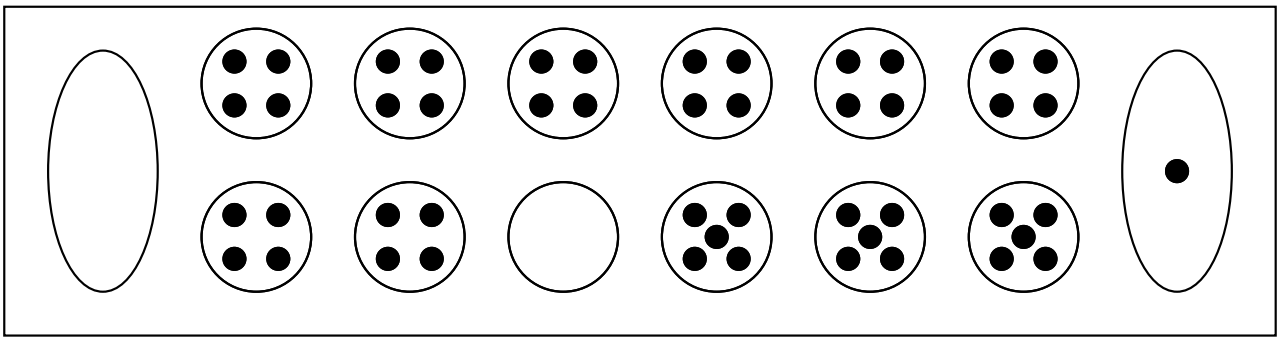
**South Player**

Initially all pits (but not store areas) start with some number of stones. In the example we give there are 6 pits per row and 4 stones per score area.

During a players turn they choose one of the pits on their side that contains at least one stone. The player takes all of the stones out of the chosen pit and distributes them in counter clockwise order, placing one stone in every pit after the current one.

If when placing stones a player encounters their store area they capture one stone from the redistribution into the score area. If the redistribution goes across the opponents store area it is ignored. If the last stone you place is in an empty pit on your own side you capture all stones in the corresponding pit on your opponents row.

**North Player**



**South Player**

The game ends when one player has no moves left, meaning all pits on their side are empty. The winner is the player who has more stones in their score area.

## 1.2 Overview

You are provided with an engine for Minimax search and alpha beta pruning. It is your task to complete the problem specification for the Mancala game and provide a useful heuristic.

All work is to be done by individual students. You may confer with other students, but you should not share your code with other students or view other students code

## 1.3 Files Supplied

All provided files use Python 3. To make use of python 3 on the cdf machines, use the command `python3` (invoking `python` will default to Python 2.7).

The following three files make up the solver you are provided:

`game.py`
`game_engine.py`
`search.py`

It is to your benefit to familiarize yourself with these files, but no alterations to them should be made for this assignment. `search.py` contains the code for doing a variant of Minimax search and alpha beta pruning. `game.py` provides an abstract base class for adversarial search problems and associated heuristics. `game_engine.py` contains code allowing for competing heuristics to be played against each other, as well as allowing a human player to play against the computer.

You will have to complete the problem description and heursitic in the file `mancala.py`

## 1.4 Submission

All submissions are to be electronic. You should submit the following files via cdf:

`mancala.py`
`a2_answers.pdf`

## 1.5 Work

In the file `mancala.py`:

1) Implement the following functions to complete the problem description

```
def __init__(self, pits, stonesPerPit):
def get_initial_state(self):
def winner(self, state):
def terminal(self, state):
def moves(self, player, state):
def next_state(self, player, state, move):
def parse_player_move(self, move_str):
def valid_move(self, state, proposed):
def print_state(self, state):
```

Moves should consist of a single integer representing the pit chosen. Pit 0 is the bottom left pit of the south player. The pits should be labeled from pit 0 in a counter clockwise order. Note that the store areas do not count as pits.

2) Implement the class `class MancalaHeuristic(Heuristic):` with your own custom heuristic.

Do not change the function header for any of the functions you implement!

As a written submission in `a2_answers.pdf` provide a short answer to the following questions:

1) Describe in your own words the heuristic you implemented.

2) Describe in your own words how you structured your states.

3) Run some tests on your heuristic for various problem sizes and search depth. How does your heuristic compare to the zero heuristic?

### 1.6 Evaluation

**[25]** Short answers

**[50]** Correct implementation

**[25]** Heuristic performance