**Mancala**

Overview: The game Mancala is a game that has many sets of rules. We will be simulating the game using rules that are easy to code. The game is played on a board that is a rectangle with a large bin at each end and 12 smaller bins in two rows in the middle.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* \* \* \* \* \* \* \* \***

**\* \* 0 \* 1 \* 2 \* 3 \* 4 \* 5 \* \***

**\* \* \* \* \* \* \* \* \***

**\* \* \* \* \* \* \* \* \***

**\* \* \*  \* \* \* \* \* \***

**\* 13 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 6 \***

**\* \* \* \* \* \* \* \* \***

**\* \* \* \* \* \* \* \* \***

**\* \* 12 \* 11 \* 10 \* 9 \* 8 \* 7 \* \***

**\* \* \* \* \* \* \* \* \***

**\* \* \* \* \* \* \* \* \***

**\* \* \* \* \* \* \* \* \***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

There are two players, Player 1 and Player 2. Each player is trying to get the most beads into their bin at the end of the board. (numbers 6 and 13)

Each player has a side and an end bin as shown by the shaded area above.

1. Each bin on the side starts with 4 beads.
   * Player 1, has bins 0 - 5 and end bin 6.
   * Player 2, has bins 7 – 12 with end bin 13.
2. A player picks up all the beads in one bin from their side and moving around the board clockwise drops a bead in each bin including their own end bin but not their opponents end bin.
3. If they run out of beads in a bin with beads, they pick up all the beads (including the one they just dropped) in the bin and continue their turn.
4. If they end in a side bin without any beads their turn is over.
5. If they end in their end bin then they get to continue their turn starting with any bin on their side of the board.
6. The game ends when one person clears their side of the board.  The player with the clear side gets all remaining beads on the board that are not in an end bin.
7. The winner has the most beads in their end bin.

Pseudocode: Pseudocode must be written and submitted on D2L along with the C++ source.

General Programming: All functions must have comments; these comments include a description of the function’s parameters, purpose and return value (if applicable…) All variables must have comments. The C++ source must have comments as well, especially for the more complicated sections.

Source Code: All the Mancala programs are cumulative, so you should build the next C++ source using the previous source code. As the homework is due in groups, e.g. Mancala 1 & 2 are due on the same date, you can submit a single source code that contains the previous functions and the new functions that are due.

Each part builds on the previous. If you have errors on one function, fix it prior to the next part as that function may be used in the next part.

Mancala Part 1:

Write a program that includes a function that outputs a line of stars named makeSolidLine(). This function is used to draw the top, middle and bottom lines of the board. The function has a single parameter for the number of stars printed out. You must use a loop. Do NOT put an endl at the end of the line in the function. If you want an endl put it where you call the function. There should no input in the code.

Test makeSolidLine() by calling it from main() with different numbers of stars.

Mancala Part 2:

Write a function, named *makeDottedLine()* that outputs a non-solid line, e.g. output a star followed by 6 spaces. **You must use nested loops,** one to output the six spaces, the other outputs the set of stars and spaces. Do **not pass** a parameter into the function and do not use setw(). Output 8 sets of stars and spaces and one extra star. Again test the function by calling it from main(). It should look like the line below:

**\* \* \* \* \* \* \* \* \***

Mancala Part 3:

Write a function called *showBoard()* that calls the functions makeSolidLine() and makeDottedLine() and produce the outline of the board as seen above; the grid/outline of the board plus bin label numbers. Make sure you are using a variable that is incremented or decremented to put the numbers on the top and the bottom. The two end bin number labels can be hard coded. You do not have to do shading. Do NOT write a single function that prints out both the top and bottom number. Write two functions *showTopRowNumbers()* to print out the top numbers and *showBottomRowNumbers()* to print out the bottom numbers. You MUST use setw and not spaces to center the numbers. It will look similar to the board in the overview.

Mancala Part 4:

Use a constant integer for the array size.

Write a function called *startingArray()* that initializes the array of beads with the value of 4 in every location except the two end bins 6 and 13, they should contain a zero. Note it is sometimes easier to fill all the bins and then reset the end bins back to zero.

The array must be declared in main and passed to functions as a parameter. Use a constant integer for the array size.

Write a function *printArray* that prints out the array. The output doesn’t have to look great but readable enough for debugging purposes. Example output for printArray() is…

4 4 4 4 4 4 0 4 4 4 4 4 4 0

Mancala Part 5:

Write a function or modify *showBoard()* so that it will also output the numbers of beads at the locations in the board. So the starting board should look like this. You will need two functions *showTopBins()* and *showBottomBins()* When testing put a different number in each location so you can see that everything is in the correct location. Hint: the two new functions will be very similar to the functions in part 3.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**\* \* \* \* \* \* \* \* \***

**\* \* 0  \* 1  \* 2 \* 3 \* 4 \* 5 \* \***

**\* \* \* \* \* \* \* \* \***

**\* \* 4 \* 4  \* 4 \* 4 \* 4 \* 4 \* \***

**\* \* \* \* \* \* \* \* \***

**\* 13 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 6 \***

**\* \* \* \* \* \* \* \* \***

**\* \* 12 \* 11  \* 10 \* 9 \* 8 \* 7 \* \***

**\* \* \* \* \* \* \* \* \***

**\* 0 \* 4 \* 4 \* 4 \* 4 \* 4 \* 4 \* 0 \***

**\* \* \* \* \* \* \* \* \***

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

Mancala Part 6:

Write a function called *gameOverCheck()* that determines if the game is over returns which player won. The game stops when one person’s middle bins are empty.  The player that empties their side first gets all of the beads left on the opponent’s side.  The winner has the most beads. The function has to check first whether the one player’s bins are empty and if so, calculate who won the game by the number of beads in the end bins. This function returns -1 if no one has won the game yet. If a player wins, it returns the player number of the winner.

Mancala Part 7:

Write a function called *getStartingBin()*, it asks the player in what bin they want to start. Its parameters include the bead array and the player number. As well, it passes the move back as reference. This function must ensures the bin is one of the specific player middle bins (not the end bin) and there are beads in the chosen bin. This function must do the checking for both player 1 and player 2.

Mancala Part 8:

Write a function *dropBeads()* that drop the beads in the bins. Its parameters include the player number to ensure the player doesn’t drop beads in the opponent’s end bin. This plays the turn to completion, e.g. pick up the beads from the bin chosen, drop beads in the next bins, if the player ends on a bin that has beads, pick up those beads and continue. If the player ends on an end bin, the player can choose a new bin to start on; make sure it’s a valid bin. The player ends on a bin with no beads, their turn is completed.

Mancala Part 9:

Modify main or write a function(s) that will let the players play. Move around the board dropping beads into the bins. Using the rules above, determine if the game is over, the player continues, or it is the other player’s turn.

Mancala 10 Extra credit:

Give the person the choice of playing against another human or against the computer. The computer has to make good choices on plays.