BBM203 ASSIGNMENT I

SUBJECT: ARRAYS

A. Atahan TÜRK

21827943

I runned my program in dev like this:

g++ -std=c++11 *.cpp -o calistir

./calistir deck.txt commands.txt output.txt

Defining Problem: We need to develop a klondike solitaire game which we can play via file I/O. We need to design a game board. Game board has different parts and each part should be represented by arrays such as foundation, stock, waste, pile. We take cards from deck.txt. 28 of cards will be in piles and 24 of them will be in stock. Closed cards in piles and stock will be represented by "@@@". We take game moves from command.txt. There 5 main command types in that file. We play the game with this file and after every command we print the new outlook of the game board in output.txt.

Explonation of my approach and classes: I had to use only arrays as dynamic data structures. So the most important thing in this assignment is using arrays efficiently. Firstly I thought "How should I use arrays? What are the things that represented by arrays?". I decided that I need an array which includes all cards(include the deck), I need an array which represent the piles, stock and waste. Also I need 4 arrays for each type of cards' foundation. First I didn't think that I need to use classes but then I recognize that it will be easy with using classes. So I create 4 classes(4.h and 4.cpp) other than main class.

Main class is for taking arguments and executing the program.

In Pile class I created arrays for piles. In this class there are functions that ordering and checking pile moves.

In Stock class I created arrays for stock and waste. Stock and waste part was the most difficult part for me. In this class there are methods that checks the stock count and updating stock and waste data accordingly, removes the right-most card from waste replacing it with ___.

In Foundation class I created 4 array for each card types' foundations. In this class there are methods that move the right-most card from the waste into foundation, checking if the foundation is empty, if empty then the card from waste must be Ace = 1, If foundation is not empty then the card from waste must be 1 higher from the top card of that foundation, moves a card from a

foundation into a pile for tactical play, checks if the game has been won using the number of cards on the foundation.

In Game class I created an array to store all 52 cards(deck) in it. In this class there are functions that reads deck file and creates a string array of cards, creates stock(waste inside of stock), pile and foundation objects, writes the game status on output file, check commands, checks if player has won.

deck[]: a string array of cards.

pile[20][7], pileSeenOnBoard[20][7]: We have 7 columns and a column can be at most 19 cards => right-most column with 6 closed cards open cards is King and it can extend to 6 + 13 cards. I used 20 for array index bounds on checks.

stock[24], waste[24], currentWaste[3]: Since 28 cards are on the pile already 24 elements are enough for stock and waste array and 3 card will be our current wastes.

hearts[13], diamonds[13], clubs[13], spades[13]: every type has 13 cards.