BLG223E – Data Structures 19.11.2018
2018–2019 Fall, Homework 3 **Updated: 30.11.2018**
Submission Deadline: 03.12.2018 23:59

Card Game

Overview

In this homework you are going to write code for a made-up card game. The game is played by two players. It is a shedding-type card game, ie. the aim is to get rid of the cards in your deck as fast as possible than the competitor. It is a turn-based game. **Your task** is to watch a gameplay, and tell how many cards are put into the bin under the table by the time game ends.

The Game

a) Two players sit around a table.

b) Each player has *equal* number of cards in their decks.

c) There is also a deck on top of the table.

d) Unlike traditional cards of standard 52-card deck, card of this game only have a positive or negative number on it. No colors, no shapes. With only a positive or negative value.

e) Each player takes one card from the table deck in his turn and according to the value of this card, he either gives or takes card to/from the other player. First player begins the game by selecting the top most card of the deck on the table.

f) If the card says a negative number, the player gives that amount of card in his deck to the competitor. In contrast, if the card was a positive number, the player takes that amount of card from the competitor.

g) When a player takes a card, he takes it from the top of the competitors deck.

h) When a player gives a card, he gives it to the top of the competitors deck.

i) However, there is one rule about taking and giving cards. Player takes cards only if the cards to be taken are *greater than* **the current card** on top of his own deck. Otherwise those cards are thrown away into the bin under the table.

j) Initially the bin is empty.

k) If a player does not have enough cards to give, he gives all his deck. This is all he can do.

l) The game ends whenever a player is out of cards in his deck, or the deck on the table is out of cards.

m) Whoever is able to get rid of his deck wins the game. If the deck on the table is out of cards instead, whoever has the least number of cards in his deck wins the game. If players have the equal number of cards in their decks when table is out of cards, the game ends in withdraw.

n) Print to the screen **a raw number, without any cosmetics (no spaces, no dots etc.)** telling how many cards there are in the bin.

Implementation

In your implementation you will read one file as *command line argument* which is formatted as:

1. The first line has two variables separated by a space: *tableDeckCount* and player*DeckCount*.
   - *tableDeckCount* defines how many cards there will be in the deck on the table.
   - *playerDeckCount* defines how many cards there will be in a players deck.
2. After the first line, you will be expecting *"tableDeckCount"* lines. In each line, there will be values of cards from bottom to top, ie. the second line is the card of the bottom of the table deck, the line "*tableDeckCount+1"* is the card of the top of the table deck.
3. After this, *through* "player*DeckCount"* lines, you will be given the deck of the first player, from bottom to top.
4. After this, *through* "player*DeckCount*" lines, you will be given the deck of the second player, from bottom to top.
5. Constraints for the variables are as follows:
   - *playerDeckCount* is an integer in range [0, 1000]
   - *tableDeckCount* is an integer in range [0, 1000]
   - a card value is an integer in range [-1000000, 1000000] except 0

Evaluation

Your homework will be evaluated in terms of your implementation. Your program should be compilable on ITU's Linux Server by the following command. Yes, you are allowed to use C++11 features in this homework if you wish!

```
>g++ -std=c++0x -Wall -Wextra -Werror main.cpp -o cardgame
```

Table 1: An example how we run your program

```
>ls
example.game   main.cpp   manyOtherInputs/
>cat example.game
1 3
-2
6
7
8
1
5
4
>g++ -std=c++0x -Wall -Wextra -Werror main.cpp -o cardgame
>./cardgame example.game
1
```

o) You read the game file name as **the command line argument** into your program.

p) You will be graded with respect to the percentage of correct output you produce for given many input game files to your implementation.

q) Although you will always be given **valid input files**, you should expect any inputs.

r) You are not allowed to include **any STL container**, such as **std::stack<T>, std::vector<T>,**

std::list<T> or others. We will change the pathway for those containers. If you use either in your implementation, your code probably will not compile. So, you will not be graded even if **your implementation used to compile** or **produce correct output**. You need to implement your own stack or other containers if you need. There are enough resource about complete implementation about these in your course slides.

s) Your homework will be evaluated by using **black-box techniques**.

Policy

- You may discuss the problem addressed by the homework at an abstract level with your classmates, but you should not share or copy code from your classmates or from the Internet. You should submit your own, individual homework.
- Academic dishonesty including but not limited to cheating, plagiarism, collaboration is unacceptable and subject to disciplinary actions.

Submission

- Please submit your file through Ninova e-Learning System.
- You must **only** submit one .cpp file named *main.cpp* Do **not** send us any other file.
- All your implementation must be in C++, and we must be able to compile and run it on ITU's Linux Server (you can access it through SSH).
- For Windows users: If you wish, you can use WinSCP to upload and edit your source code into ITU SSH Server, and use PuTTY to compile and run your algorithm. If does not, please make sure that your code is able to **be compiled** and runned on ITU's Linux Server.
- Your code **has to be compiled successfully**. Otherwise you will not be graded. Which will result a grade of **zero** for the homework.
- You should be aware that the Ninova e-Learning System clock may not be synchronized with your computer, watch, or cell phone. Do not e-mail the teaching assistant or the instructors about your submission after the Ninova site is closed for submission. If you have submitted to Ninova once and still want changes in your report, you should do this before the Ninova submission system closes. Your changes will not be accepted by e-mail. Connectivity problems in the last minute about the internet or about Ninova are not valid excuses for being unable to submit. You should not risk your project by leaving its submission to the last minute. After uploading to Ninova, make sure that your homework appears there.

START EARLY.

If a point is not clear, discuss it or e-mail kcengiz@itu.edu.tr

Examples

Updated: 4 more examples (examples.zip) were added to Ninova.

1 Example 1

This example prints 1 to screen.

Input file	How it works
1 3 -2 6 7 8 1 5 4	--turn 1 begins Cards of table: [-2] Cards of the first player: [6 7 8] Cards of the second player: [1 5 4] First player need to give 2 cards 8 > 4 is true Cards of the first player: [6 7] Cards of the second player: [1 5 4 8] 7 > 8 is false Card 7 goes to bin under table Cards of the first player: [6] Cards of the second player: [1 5 4 8] --turn2 begins Cards of table: none Cards of the first player: [6] Cards of the second player: [1 5 4 8] Game ends. First player wins. Cards of the bin: [7]

2 Example 2

This example prints 1 to screen.

Input file	How it works
2 3 -1 -2 1 4 5 6 9 3	--turn 1 begins Cards of table: [-1 -2] Cards of the first player: [1 4 5] Cards of the second player: [6 9 3] Cards of bin: [] First player needs to give 2 cards 5 > 3 is true → Card 5 is given 4 > 5 is false → Card 4 is thrown away --turn 2 begins Cards of table: [-1] Cards of the first player: [1] Cards of the second player: [6 9 3 5] Cards of bin: [4] Second player needs to give 1 cards 5 > 1 is true → Card 5 is given Game ends, table is out of cards. First player wins. There are 1 cards in the bin.