Programming assignment #1

Stack & Queue

Objective

1. To understand how an implementation of an ADT is used by an application program.
2. To become familiar with how to implement the Stack & Queue .

Problem Definition

There are a *stack* of *deque* and a *queue* for saving *deque* temporarily. You have to store piles of cards as *deques* and put them into a *stack* in order.

After getting data, you have to number the stack from top to bottom starting with 1 and ending with its size. We then give you two different indices, you have to take them from the stack and put the unused ones into queue one after another.

Of the two *deque*s, you take the first card of the second index *deque* comparing with the first card of first *deque*. Cards match if they are of the same suit or same rank, put second one before the first card of first *deque*; otherwise, put it as the last. Each moving causes the second *deque* decreasing by one card. Repeating the steps above, the second *deque* has no more cards, and there is one combining *deque*. Put the *deque* in *queue* according to the last card being first or not. Nevertheless, put the *deque* from *queue* first and then the combining one.

After several iterations, there is only one *deque* in the *stack*. File out all cards starting from top of the *deque*.

I/O Format

Your program must be able to read an input file. The I/O file names are arguments to your program; in other words, those file name can NOT be fixed, referencing the attached. In command line, your program will be invoked by:

```
./a.out  input_file_name  output_file_name
```

Input data contains irregular quantity of cards separated by single space characters. The input of cards ends up with “#” character. Cards are represented as a two characters code. The first character is the face-value (A=Ace, 2–9, T=10, J=Jack, Q=Queen, K=King) and the second character is the suit (C=Clubs, D=Diamonds, H=Hearts, S=Spades). Every card is distinct, so total quantity of cards is up to 52.

Two indices of *deques* for the *stack* will be given after reading all cards.

Input file example

```
8H JC
KC
QH 2C 8D JS
#
1 3
2 1
```

Output all elements in the *deque* of *stack*.

Output file example

```
JS JC KC 8D 2C QH 8H
```

Program Submission

1. Please use C/C++ language and your program **must** be written in **only one** source file.
2. Your source file must be named as "Student_ID_number_pa1.cpp" and please make sure that all characters of the filename are in **lower case**. For example, if your student number is 0510101, the name of your program file should be "0510101_pa1.cpp".

Report

1. Your report must contain the flow chart or the pseudo code of you program. You have to describe how your approach works. The analysis of time complexity should be included as well.
2. The report file name must be "Student_ID_number_pa1.pdf" and please make sure that all characters of the filename are in lower case. For example, if your student number is 0510101, the name of your program file should be "0510101_pa1.pdf".

Grading

You need to submit both your source code and report. Remember the submission rules mentioned above, or you will be punished on your grades.

- Unique and compilable source code 30 %
- Five cases 50 %
- Report 20 %

Due Date

Upload your report and program to the e3 platform. All of your files must be archived to only one file named "**Student_ID_number_pa1.zip**" or "**Student_ID_number_pa1.rar**". You have also to hand in the "*Honesty Declaration*" on class on October 17, or it will be scoreless on programming assignment 1.

The upload deadline is **23:59 October 16**.