Assignment 2

Seon Joo Kim
Yonsei University

Outline

- Deadline
- No Plagiarism
- Leave Comments
- Scoring
- Problem 1
- Problem 2
- Problem 3
- Problem 4
- Submission
- Questions

Deadline

- Tuesday, October 6th 23:55
- No late submissions at all

No Plagiarism

- No Mercy.

- The punishment will be made to both
  - the person who copied the code, and the person who shared the code.

- We will do plagiarism test with codes that were made in previous semesters and also in google. So be careful ☺

Leave Comments

- Leave comments in your file for TAs to understand your code.
- If no comments in the file, there may be a reduction of points.

Scoring

- You should take care of your code not terminating by an issue in the middle of the loop
  - Scores will be given only by the final outputted file

- Example
  - 5 test cases
  - If your code is correct as O X O O O if ran seperately

  but terminates in the second test case by an error

  only the first test case is considered correct

- **Problem1 (15%)** **Problem2 (25%)**
- **Problem3 (30%)** **Problem4 (30%)**

- If your code outputs correctly for given example input#.txt file
  - 30% base score per problem
- There will be additional 20 test cases
  - (70 / 20) = 3.5% per each case

Scoring

- When ran the code, the printed text *(by cout)* in the terminal can be recorded in a file by '>>' command
- We will score the results by saving your programs printed texts by '>>', and compare by 'diff' command
- problem1 ~ problem4 will be graded in this way.
- Example

```
$ g++ -Wall problem1.cpp –o problem1
$ ./problem1 >> output1.txt
$ diff answer.txt output1.txt
```

Problem 1

- Write a program that reads an array of characters in English and counts how many vowels are included.

- You have to use arrays to save input and counts of vowels.

Problem 1

Since we have not yet learned much about c++ string, I'll give you a **hint**.

1. '\0'(null character) is a special character that automatically attaches to the end of a string.
2. you could use cstring library to handle string in c++. Ex. strlen

added on 200924

- Instruction
  - Input
    - First row of input is number of loops : N
    - From the second row, each line contains array of characters.
    - Assume that there is no input more than 50 characters.
    - No error case for this input, so don't need to handle exception for this input.
    - Only small letter alphabets and space are allowed for input. (Do not care about capital letter)
  - Output
    - Each line contains the number of occurrences of each vowel in **a, e, i, o, u order** (alphabetic order).
- I/O example

Input (input1.txt)	Output
3 hello world yonsei university object oriented programming	0 1 0 2 0 0 2 3 1 1 1 3 2 3 0

→ a:0 e:1 i:0 o:2 u:0

→ a:0 e:2 i:3 o:1 u:1

→ a:1 e:3 i:2 o:3 u:0

Problem 2

- Write a last-bit restoration program using parity bit algorithm.

- A **parity bit**, or **check bit**, is a bit added to a string of binary code to ensure that the total number of 1-bits in the string is even or odd. Parity bits are used as the simplest form of error detecting code.

- odd parity is a bit-string composed of odd number of 1's. and even parity is a bit-string composed of even number 1's. (the bit-string without 1 is also even parity)

- If the last digit in an input bit-string is removed, make a program for input restoration, given a parity bit.

- Example

  Input: 110010_2 need 1 to be even parity case

  Even parity case: 1100101_2 → 101(Results) (64+32+4+1=101)

  Odd parity case: 1100100_2 → 100(Results) (64+32+4=100)

Problem 2

- Instruction
  - Input
    - First row of input is number of loops : N
    - From the second row, each line contains information for parity restoration.
    - binary number length (space) Binary number (space) Even parity: e, odd parity: o
    - binary number length: 0 < x < 50, # There are ‘spaces’ between binary number digits.
  - Output
    - Each line contains restored input which is converted to decimal.

- I/O example

Input (input2.txt)	Output
4 6 1 1 0 0 1 0 e 6 1 1 0 0 1 0 o 3 1 1 1 e 1 0 e	101 100 15 0

Problem 3

- Write a program for ***Bigger is better*** game

- Instruction for ***Bigger is better***
  - This is a simple game. There are 2 players(A, B) in this game, and they are given N cards when the game started. They have to open the cards in order where they are given. When they open their cards, they compare them and one who has a bigger card becomes the winner. Loser's card is discarded, and winner's card is degraded by the difference of cards. If the cards' number is the same, they draw(no winner), and the cards of both of the players are discarded. The winner of each ground takes 1 pts, and the game is over if one player has no card more. The simulation of the game is introduced on the next slide.

Problem 3

<Example>

Initial State:	**Ground 1:**	**Ground 1:**	**Ground 2:**	**Ground 2:**
Player A (score : 0) Cards : 6 / 1 / 4	Player A (score : 1) Cards : 6 / 1 / 4 Open 6 ➔ Win	Player A (score : 1) Cards : 2 / 1 / 4 6 – 4 = 2(degraded)	Player A (score : 1) Cards : 2 / 1 / 4 Open 2 ➔ Draw	Player A (score : 1) Cards : 1 / 4 Discard 2
Player B (score : 0) Cards : 4 / 2 / 6	Player B (score : 0) Cards : 4 / 2 / 6 Open 4 ➔ Lose	Player B (score : 0) Cards : 2 / 6 Discard 4	Player B (score : 0) Cards : 2 / 6 Open 2 ➔ Draw	Player B (score : 0) Cards : 6 Discard 2
Ground 3:	**Ground 3:**	**Ground 4:**	**Ground 4:**	**Final State:**
Player A (score : 1) Cards : 1 / 4 Open 1 ➔ Lose	Player A (score : 1) Cards : 4 Discard 1	Player A (score : 1) Cards : 4 Open 4 ➔ Lose	Player A (score : 1) Cards : Discard 4	Player A (score : 1) Cards :
Player B (score : 1) Cards : 6 Open 5 ➔ Win	Player B (score : 1) Cards : 5 6 – 1 = 5(degraded)	Player B (score : 2) Cards : 5 Open 5 ➔ Win	Player B (score : 2) Cards : 1 5 – 4 = 1	Player B (score : 2) Cards : 1

Problem 3

- Instruction
  - Input
    - First row of input is number of loops : N
    - From the second row, each three line contains information for ***bigger is better*** game.
    - The number of cards (K) is given in the first row (1 ≤ **K** ≤ 1000)
    - **K** card number (**c**) streams in the next 2 rows (0 ≤ **c** ≤ 10000) – always integer
  - Output
    - Each line contains scores of each player in each game
    - output format → player1's_score (space) player2's_score

- I/O example

Input (input3.txt)	Output
3 3 6 1 4 4 2 6 5 9 10 6 12 7 6 1 5 11 20 7 10 2 9 33 17 5 21 6 19 33 18 3 0 11	1 2 5 4 6 5

Problem 4

- Write a program for ***airline reservation system.***
  - ***tip : Call-by-reference practice***

- Our system should provide these two features.
  1. add passenger
  2. delete passenger

- There are three types of seats and each type has different number of seats.
  1. first class (total seats: 5)
  2. business class (total seats: 25)
  3. economy class ((total seats: 50)

- At first, all flight seats are vacant.

Problem 4

1. add passenger (use *void add_passenger*)

- Instruction
  - Function to reserve seats
  - If the number of seats to be reserved is more than limit, the number of passengers remains unchanged and print “full seat”
  - If adding passenger is complete, print “add complete”

Problem 4

2. delete passenger (use *void delete_passenger*)

- Instruction
  - Function to cancel reservation
  - If the number of seats to be deleted is ~~less~~ **more** than reserved seat, the number of passengers remains unchanged and print “wrong input”
  - If adding passenger is complete, print “delete complete”

revised on 200924

Problem 4

3. show seat state (use *void print_seat_state*)

- Instruction
  - Function to show availability map and number of remaining seats
  - Seat available: O (capital O), unavailable: X (capital X)

- Examples

```
current vacant seats
first class: 5/5
business class: 25/25
economy class: 50/50
first class
OOOOO
business class
OOOOO
OOOOO
OOOOO
```

```
OOOOO
OOOOO
economy class
OOOOO
OOOOO
OOOOO
OOOOO
OOOOO
OOOOO
OOOOO
OOOOO
OOOOO
OOOOO
```

```
current vacant seats
first class: 2/5
business class: 15/25
economy class: 0/50
first class
XXXOO
business class
XXXXX
XXXXX
OOOOO
```

```
OOOOO
OOOOO
economy class
XXXXX
XXXXX
XXXXX
XXXXX
XXXXX
XXXXX
XXXXX
XXXXX
XXXXX
XXXXX
```

Problem 4

- Instruction
  - Input
    - First row of input is number of function calls : N
    - From the second row, each line contains information of two types of features
    - feature_type (space) seat_type (space) number_of_passengers

feature type
1 → add_passenger
2 → delete_passenger

seat type
1 → first class
2 → business class
3 → economy class

do not consider about other inputs
Ex.
~~feature_type = 3~~
~~seat_type = 4~~

added on 200924

  - Output
    - Cout when each event occurs, and when function calls are end, print current passenger state.

Problem 4

- I/O example.

(2/5) (25/25) (50/50)
(2/5) (25/25) (50/50)
(2/5) (0/25) (50/50)
(2/5) (0/25) (50/50)
(2/5) (11/25) (50/50)

Input (input4.txt)	Output	
5 1 1 3 1 3 55 1 2 25 2 3 30 2 2 11	add complete full seat add complete wrong input delete complete current vacant seats first class: 2/5 business class: 11/25 economy class: 50/50 first class XXXOO business class XXXXX XXXXX XXXXO OOOOO OOOOO economy class OOOOO (1)	OOOOO OOOOO OOOOO OOOOO OOOOO OOOOO OOOOO OOOOO OOOOO end the program (2)

Submission

- Zip the folder by following steps correctly

```
oop@oop-VirtualBox:~/Desktop$ cd 2020123456_hw2/
oop@oop-VirtualBox:~/Desktop/2020123456_hw2$ ls
problem1.cpp  problem2.cpp  problem3.cpp  problem4.cpp
oop@oop-VirtualBox:~/Desktop/2020123456_hw2$ cd ../
oop@oop-VirtualBox:~/Desktop$ tar -zcvf 2020123456_hw2.tar.gz 2020123456_hw2/
2020123456_hw2/
2020123456_hw2/problem3.cpp
2020123456_hw2/problem4.cpp
2020123456_hw2/problem2.cpp
2020123456_hw2/problem1.cpp
```

- studentId_hw2.tar.gz
  - Ex) 2020123456_hw2.tar.gz
- There is going to be reduction of points if not following the folder hierarchy as well
- If unzipped your submission .tar.gz file should follow the folder hierarchy below

Current directory

- studentId_hw2.tar.gz
- studentId_hw2
  - problem1.cpp
  - problem2.cpp
  - problem3.cpp
  - problem4.cpp

Questions

- Use oop20202@gmail.com for questions

- We are not going to answer
  - Questions sent to TAs' personal mails
  - Questions not making sense
  - Questions related to the algorithm for solving the question
  - Questions you can infer the answer if read this file thoroughly
  - Questions you can simply solve by googling
    - Ex) how do I make a folder on ubuntu?

Appendix

- File I/O

```
#include <fstream>
ofstream outfile;
outfile << "Hello, World!\n"; // writing Hello,World! into the file

outfile.close(); // should close the file before terminating the process

ifstream infile("input.txt");
infile >> number; // reading the first digit written in input.txt

infile.close(); // should close the file before terminating the process
```

https://stackoverflow.com/questions/7868936/read-file-line-by-line-using-ifstream-in-c

Appendix

- Zipping and unzipping the folder by tar command
  - https://linuxize.com/post/how-to-extract-unzip-tar-gz-file/
  - https://www.cyberciti.biz/faq/how-do-i-compress-a-whole-linux-or-unix-directory/