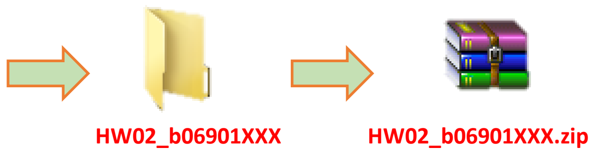
Computer Programming Homework #2

2017/12/11 by TA 陳泓弦 陳姿玲

REQUIRED FILES

Please **compress a folder** named **HW02_b06901XXX** (student ID) that contains the following files:

- ✓ b06901XXX _p1 project
- ✓ b06901XXX _p2.cpp
- ✓ b06901XXX _p3.cpp
- ✓ b06901XXX _p4.cpp
- ✓ b06901XXX _bonus.cpp



Do not submit executable files (.exe). Files with names in the wrong format will not be graded. In your .cpp files, we suggest you write comments in details as much as you can. It will be good for TAs to read your code and for your future reference and maintenance.

Homework #2 is due on December 25, 2017 .

PROBLEM DESCRIPTION

Problem 1. [25 points] [Require: b06901XXX _p1 project]

Design a program that reads a positive number A and executes the following steps:

Step1: Form a new number B by sorting each digit of A in decreasing order.

Step2: Form a new number C by sorting each digit of A in increasing order.

Step3: Output A, where A=B−C .

Step4: Repeat Step1 to Step3 and output A until it has appeared in previous rounds

For example : input a number A =94510

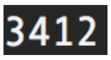
```
95410(number B) – 1459(number C) = 93951
99531 - 13599 = 85932
98532 - 23589 = 74943
97443 - 34479 = 62964
96642 - 24669 = 71973
97731 - 13779 = 83952
98532 - 23589 = 74943
Output: 7
```

Note:

- ✓ You can use any function to sort digits of A in each round.
- ✓ The maximum A for input is 99999.
- ✓ The program does not stop until you input “0” for number A.
- ✓ You should define all functions you need in sortlib.cpp and declare them in sortlib.h file.
- ✓ In this problem you need to submit main.cpp, sortlib.cpp, sortlib.h in a folder named “b06901XXX_p1”

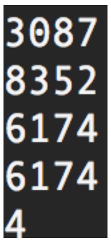
Format:

Input: (Number)



```
3412
```

Output: (A in Step3 in each round, total rounds)



```
3087
8352
6174
6174
4
```

Then input another number A again until “0” is given to stop the program.

[Bonus] (5 points)

By following the second problem’s steps in PSA09, create the static library “sortlib.a” that consists of all functions needed and link the library in your main file. Please also submit the **sortlib.a** in the folder “b06901XXX_p1”. If you fulfill problem 1 and the bonus, you will get total 30 points.

Problem 2. [25 points] [Require: b06901XXX _p2.cpp]

A child wants to go up a staircase with n steps, and he can hop up 1 or 2 steps at a time. How many different ways can the child reach the top? Please write a program to read the input *n* as the staircase steps, and write a function **countWays(int *n*)** that returns an integer as the number of different ways to reach the top.

Hint: The child can reach nth stair step from either (n-1)th stair or from (n-2)th stair

Example: If n = 4, there are total 5 different ways to reach the top.

(1st step, 2nd step, 3rd step, 4th step),

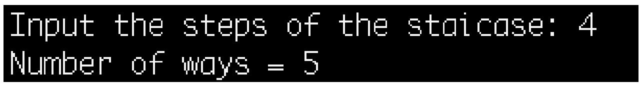
(1st step, 2nd step, 4th step)

(2nd step, 3rd step, 4th step),

(1st step, 3rd step, 4th step)

(2nd step, 4th step)

Format:



```
Input the steps of the staicase: 4
Number of ways = 5
```

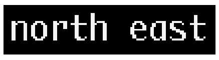
Problem 3. [25 points] [Require: b06901XXX _p3.cpp]

Let's play a game using a die. A die is a cubic object with six faces, each of which represents a different number from one to six. For simplicity, assume that the die tumbles in four directions: north, east, south, and west. At the beginning of the game, we put the die on the table and adjusts its orientation so that the numbers one, two, and three are seen on the top, north, and west faces, respectively. Note that the sum of the numbers on any pair of opposite faces is always seven. Please write a program that accepts a sequence of commands, each of which is either "north", "east", "south", or "west". A "north" command tumbles the die down to north, that is, the top face becomes the new north, the north becomes the new bottom, and so on. Other commands also tumble the die accordingly to their own directions. Your program should output the number finally shown on the top after performing the commands in the sequence.

We will input a command sequence which consists of one or more command that is one of 'north', 'east', 'south', and 'west' and use a white space to separate every command. The command sequence includes at most 10 commands. In this problem, you can use **cin.getline()** to read the input sequence and use **strtok** to deal with the white blank.

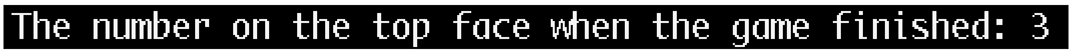
Format:

Input:



```
north east
```

Output:



```
The number on the top face when the game finished: 3
```

Hint for cin.getline(char* s, int n):

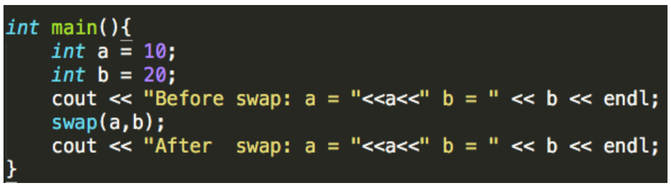
Extracts characters from the stream as unformatted input and stores them into *s* as a c-string, until either the extracted character is the delimiting character ('\n'), or *n* characters have been written to *s* (including the terminating null character).
For detail description of cin.getline(), you can google it with articles like:
https://kknews.cc/zh-tw/other/89458yg.html

Hint for strtok(char* str, const char* delimeters):

A sequence of calls to this function split *str* into tokens, which are sequences of contiguous characters separated by any of the characters that are part of *delimiters*.
To determine the beginning and the end of a token, the function first scans from the starting location for the first character **not** contained in *delimiters* (which becomes the beginning of the token). And then scans starting from this beginning of the token for the first character contained in *delimiters*, which becomes the end of the token. The scan also stops if the terminating null character is found. This end of the token is automatically replaced by a null-character, and the beginning of the token is returned by the function.
For detail behavior, you can watch it with debugger or articles like :
https://cg2010studio.com/2011/07/10/strtok/

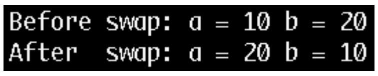
Hint: You could use *swap*() to deal with the action of tumbling a die.

Example:



```
int main(){
    int a = 10;
    int b = 20;
    cout << "Before swap: a = "<<a<<" b = " << b << endl;
    swap(a,b);
    cout << "After  swap: a = "<<a<<" b = " << b << endl;
}
```

Output:



```
Before swap: a = 10 b = 20
After  swap: a = 20 b = 10
```

Problem 4. [25 points] [Require: b06901XXX _p4.cpp]

Use SDL to design a program based on the following rule :

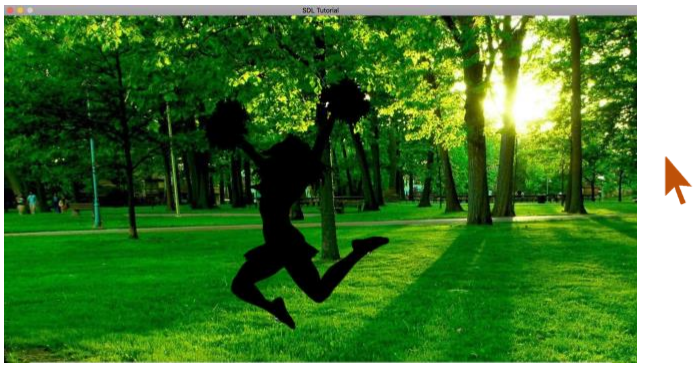
Rule1: Show "dance1.png" in the screen when the screen is untouched by the cursor

Rule2: Show "dance2.png" in the screen when the screen is touched by the cursor

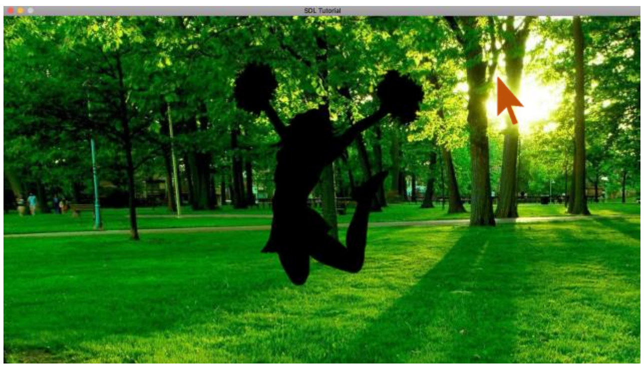
Rule3: Show "dance3.png" in the screen when the cursor moves on the screen

For example:

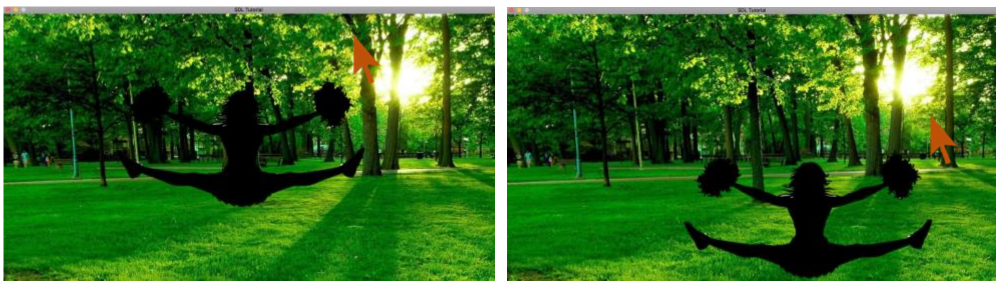
(The cursor untouched the screen)



(The cursor touched the screen)



(The cursor has touched and move on the screen)



Note:

- ✓ You need to load the "background.jpg" as the background of the screen.
- ✓ The path of picture is "image/xxx".

(Hint: You can find the sample on: http://lazyfoo.net/tutorials/SDL/index.php)

[Bonus]. [25 points] [Require: b06901XXX _bonus.cpp]

Do you know that you can play Snake game (Easter Egg) on youtube while waiting film for preparing? You can easily fine films like this https://www.youtube.com/watch?v=rp4Z27C23JQ to be familiar with this game. After learning pdcurses and C++ array controls, all of us can write an easy version of this game. Please implement it in the snake.cpp. We pre-define some variables that you may need, you can use them to fulfill snake.cpp. You can use your own variable instead.

For our convenience, here are some rules that you need to follow and make this problem easier to implement and grade:

(1) The map (environment) our snake wandering is 15(Width)x10(Height). And the map is surrounded by walls: ‘#’. (No extra wall anymore)

(2) The initial length of our snake is 4. Starting from (1,1)(tail) to (1,4)(head), with initial direction of heading right. (use ‘@’ to denotes snake)

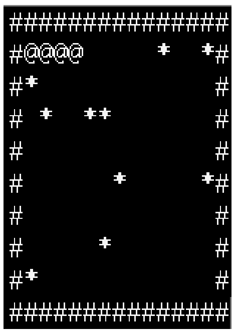
(3) User can control the heading direction by keyboard. But if user doesn’t press control keys, keep the moving direction. (The snake would always move one step per 300ms.)

(4) As for the food (egg), uses ‘*’. When the snake get the food, the length of it increase 1. (The head replaces the egg while the tail keeps)

(5) There are 10 eggs in the map. When the snake get an egg, you will get 10 points. After having 10 eggs, exit the game and show “ You win!!!”. If the snake failed, exit the game and show “DEAD! You get xx points.”

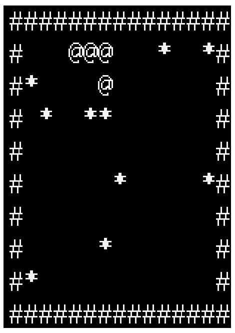
(6) For your convenience, you can place food in advance, without considering where to place or random generate food.

There’s an example for you:



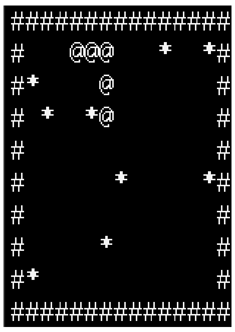
```
###############
#@@@@     *  *#
#*            #
# *  **       #
#             #
#      *     *#
#             #
#     *       #
#*            #
###############
```

→



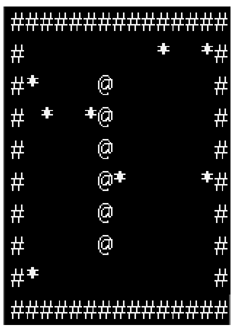
```
###############
#   @@@   *  *#
#*    @       #
# *  **       #
#             #
#      *     *#
#             #
#     *       #
#*            #
###############
```

→



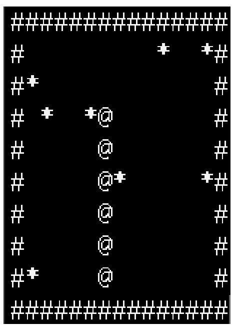
```
###############
#   @@@   *  *#
#*    @       #
# *  *@       #
#             #
#      *     *#
#             #
#     *       #
#*            #
###############
```

→



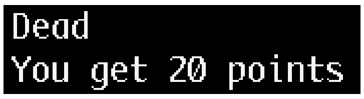
```
###############
#         *  *#
#*    @       #
# *  *@       #
#     @       #
#     @*     *#
#     @       #
#     @       #
#*            #
###############
```

->



```
###############
#         *  *#
#*            #
# *  *@       #
#     @       #
#     @*     *#
#     @       #
#     @       #
#*    @       #
###############
```

Output:



```
Dead
You get 20 points
```