

Introduction to Computers and Programming LAB-Quiz3

2016/12/28 Time: 2 hrs

※ Please create a new folder and name it as your Student ID. Inside the folder, your file format will be Q_1.c, Q_2.c, etc. (There will be score deduction for wrong file name).

※ Any C library is forbidden, except for <stdio.h>, <stdlib.h> and <string.h>

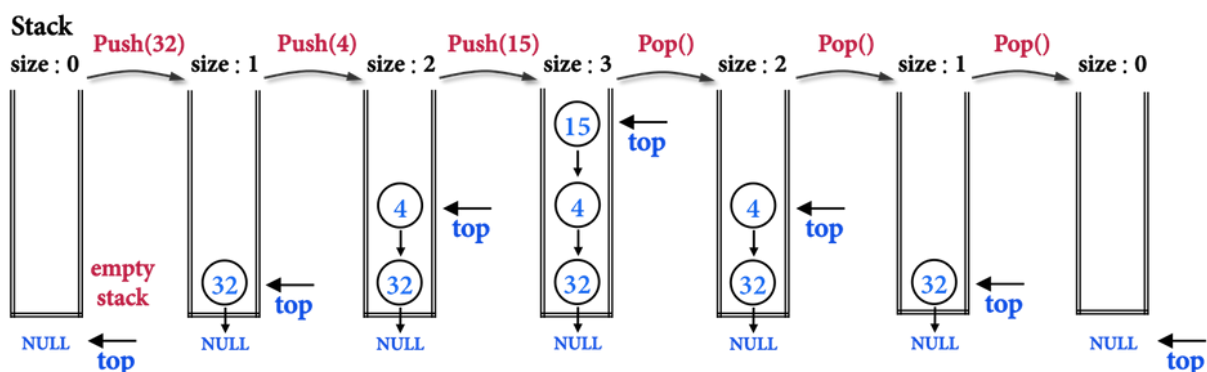
※ We will ask you to leave and give you 0 point if any suspicious or cheating behavior is found

※ The class is for C language, so do not use C++.

※ If your program cannot be compiled, you will get zero point for the question.

※ Follow the input/output format in examples.

1. Linked Stack



Source: goo.gl/vu4OJO

A stack can be implemented in different data structures, such as a static array, a dynamic array and even a linked list. **Implement a linked list version of an integer stack in this question.** The program first asks users to input the number of function: **(1) Push**: ask users to input the value of the pushed node, then push the node to the top and show the stack from the top; **(2) Pop**: pop/delete the first node and show the stack from the top; **(3) Exit**: delete all the nodes in the linked stack and exit the program. We will check whether there is any memory leak in your program, and **any array declaration/indexing is forbidden.**

```
(1) Push (2) Pop (3) Exit
Enter the number of the function : 1
Enter the value to be pushed: 32
The list content: 32 -> NULL
```

```
(1) Push (2) Pop (3) Exit
Enter the number of the function : 1
Enter the value to be pushed: 4
The list content: 4 -> 32 -> NULL
```

```
(1) Push (2) Pop (3) Exit
Enter the number of the function : 2
The list content: 32 -> NULL
```

```
(1) Push (2) Pop (3) Exit
Enter the number of the function : 2
The list content: NULL
```

```
(1) Push (2) Pop (3) Exit
Enter the number of the function : 2
The list content: NULL
```

```
(1) Push (2) Pop (3) Exit
Enter the number of the function : 3
Bye!
```

Hint:(1) Push: add a node to the first of the linked list

(2) Pop: delete the first node.

2. Auto Scoring Solutions

Write a C program that searches through a list file and then evaluate score files according to answer file and output result to a file. Here is the file formats:

answer.txt:

The first line is the standard answer of selection (total 10 questions)

The second line is the standard answer of calculation (split by space, total 10 questions)

The total score is 100, **5pts for each question.**

fileList.txt:

In this file, each line is a **filename** of **student answer file**

STUDENT_ANSWER_FILE (file names in fileList.txt):

The first line of this file is student id.

The second line of this file is gender

The third line is the student answer of selection (total 10 questions)

The fourth line is the student answer of calculation (split by space, total 10 questions)

First, the program will read "fileList.txt" which contains many file names.

For each file name in "fileList.txt", try to open it

If the file name is invalid

print "Error happened: Cannot read **INVALID_FILE_NAME_HERE** !!\n"

else

evaluation the student answer

output student id, gender and score to **result.txt**

(**Output format:** `fprintf(result, "%s %c\t%d\n", ID, Gender, Score);`)

At the end, print how many file your program processed successfully.

Example: (the test data in q2_testData)

```
Error happened: Cannot read 123.txt!!
Processed 3 file successfully
```

<<In result.txt>>

0456059 M 100

0123456 F 90

0052456 M 50

Note:

(1) We will put your C program, answer.txt, fileList.txt, and the student files **in same directory**.

(2) student id and file name length **will not exceed 100 characters**

Hint:

(1) `fscanf(fp, " %c", &c);` means **ignore space, '\t', '\n' or other blank characters**

3. Format Blocks

Design a program that reads from a file, and writes to files according to the file formats below, respectively.

The format of the input file: It contains several **blocks**, which may be separated by any number of empty lines (can be zero empty lines). The first line of each block is started with a “>” sign as the first character of the line, followed by a **FileName**, and followed by several **tags**, separated by spaces. The other lines are the **data**.

The format of the output files: Your program should generate an output file for each **block**, titled by **FileName.txt**. The file should first contain the **tags**, each in a line; and then the **data**, which is **concatenated in only one line**.

Functions in **data** area:

1. If ‘-’ is the first character of the line, the **data** in the file block before the ‘-’ need to be abandoned.
2. If number (0-9) is the first character of the line, then the data in that line need to be written to the file as the number times.
3. Otherwise, just write the data into files as normal.

Note: Each line wouldn’t be longer than 1000 characters.

```
Please enter the input file name:111.fa
No such file existed.
Please enter the input file name:2.fa
finished
Please enter the input file name:
```




Input file

```
2.fa - 記事本
檔案(F) 編輯(E) 格式(O) 檢視(V) 說明(H)
>ACX53679 Avian 2009 H7N9
MERIK
2NEOG
IR
LVRKTRF

>ADN34723 Hunan 2009
ME
NEOQQ
IRRRVDINPGHA
LVRKTRF

>AFH41223 Avian      2011 H3N8 5566
MERIKELR

NEOQQTLWSKTNDA
```

Output files		
 ACX53679 - 記事本 檔案(F) 編輯(E) 格式(O) 檢視 Avian 2009 H7N9 MERIKNEQGNEQIRLVKTRF	 ADN34723 - 記事本 檔案(F) 編輯(E) 格式(O) Hunan 2009 IRRRVDINPGHALVRKTRF	 AFH41223 - 記事本 檔案(F) 編輯(E) 格式(O) 檢視 Avian 2011 H3N8 5566 MERIKELRNEQGQTLWSKTNDA

4. Inversion Vector and Permutation

Let $\mathbf{a}[0,1,\dots,n-1]$ be an array that contains a permutation of the integers $0, 1, 2, \dots, n-1$. Define an array $\mathbf{b}[0,1,\dots,n-1]$ by:

$\mathbf{b}[k]$ = the number of values in $\mathbf{a}[0,1,\dots,k-1]$ that are less than $\mathbf{a}[k]$

The array \mathbf{b} is called the inversion vector of the permutation.

For example, let $n=10$, and the permutation be:

$\mathbf{a}[10]=\{2, 4, 7, 6, 0, 9, 1, 8, 5, 3\}$

Then, the inversion vector is:

$\mathbf{b}[10]=\{0, 1, 2, 2, 0, 5, 1, 6, 4, 3\}$

Define “The next higher permutation of an array”:

The next higher permutation of an array is the next permutation of the elements in the array in lexicographical order.

The lexicographical order:

$[a_1, \dots, a_n] < [b_1, \dots, b_n]$

if either

$a_1 + \dots + a_n < b_1 + \dots + b_n$

or

$a_1 + \dots + a_n = b_1 + \dots + b_n$ and $a_i > b_i$ for the greatest i such that $a_i \neq b_i$

Example 1. $a[3]=\{1,0,2\}$

All permutations of the elements in array **a** in lexicographical order are:

0	1	2
0	2	1
1	0	2
1	2	0
2	0	1
2	1	0

← Array **a** now

← The next higher permutation of array **a**

The next higher permutation of **a** is: $\{1,2,0\}$ and the next is $\{2,0,1\}$

Example 2. $a[5]=\{11,17,6,9,1\}$

The next 5 higher permutations of array **a** are: $\{11,17,9,1,6\}$, $\{11,17,9,6,1\}$, $\{17,1,6,9,11\}$, $\{17,1,6,11,9\}$, $\{17,1,9,6,11\}$.

Your job is to design a program that can construct the permutation (Array **a**) from its inversion vector (Array **b**). Then calculate the next *m* higher permutation of Array **a**. And then, print Array **a** and all the next m higher permutation of Array **a**.

Note: If there is no next higher permutation of Array **a**, just print the highest permutation.

```
Please input n:10
Please input elements of the inversion vector:
0 1 2 2 0 5 1 6 4 3

Array a is:2,4,7,6,0,9,1,8,5,3

Please input m:3
The next 3 higher permutations of Array a are:
2,4,7,6,0,9,3,1,5,8
2,4,7,6,0,9,3,1,8,5
2,4,7,6,0,9,3,5,1,8
```

```
Please input n:10
Please input elements of the inversion vector:
0 0 0 0 0 0 0 0 0 0

Array a is:9,8,7,6,5,4,3,2,1,0

Please input m:5
The next 5 higher permutations of Array a are:
9,8,7,6,5,4,3,2,1,0
9,8,7,6,5,4,3,2,1,0
9,8,7,6,5,4,3,2,1,0
9,8,7,6,5,4,3,2,1,0
9,8,7,6,5,4,3,2,1,0
```