자료구조 과제(HW2)

전공: 철학과 학년: 3학년 학번: 20180032 이름: 남기동

1. pmatch_all()

<Pseudo Code>

Variable: string, pattern, failure

Function:

1. void fail(char* pattern, int* failure)

Put -1 to the first index of failure(it means nothing match from the beginning)

Search pattern from 1 to the length of pattern

While) next pattern doesn't match, move to the point where it matches and find whether it match or not

If) If the same pattern matches in the next, put the index how long the pattern matches to failure

Else) put -1 to failure

2. void pmatch_all(char* string, char* pattern, int* failure)

Find whether each character in string and pattern match

If) pattern match, move to next character until pattern is over

If all pattern is matched and pattern is over,

then return start point where the pattern and string are matched

else if) not match at the first character of pattern, only move next in string

else) move pattern to the point that indicates where the values match from the beginning of the pattern

3. main function

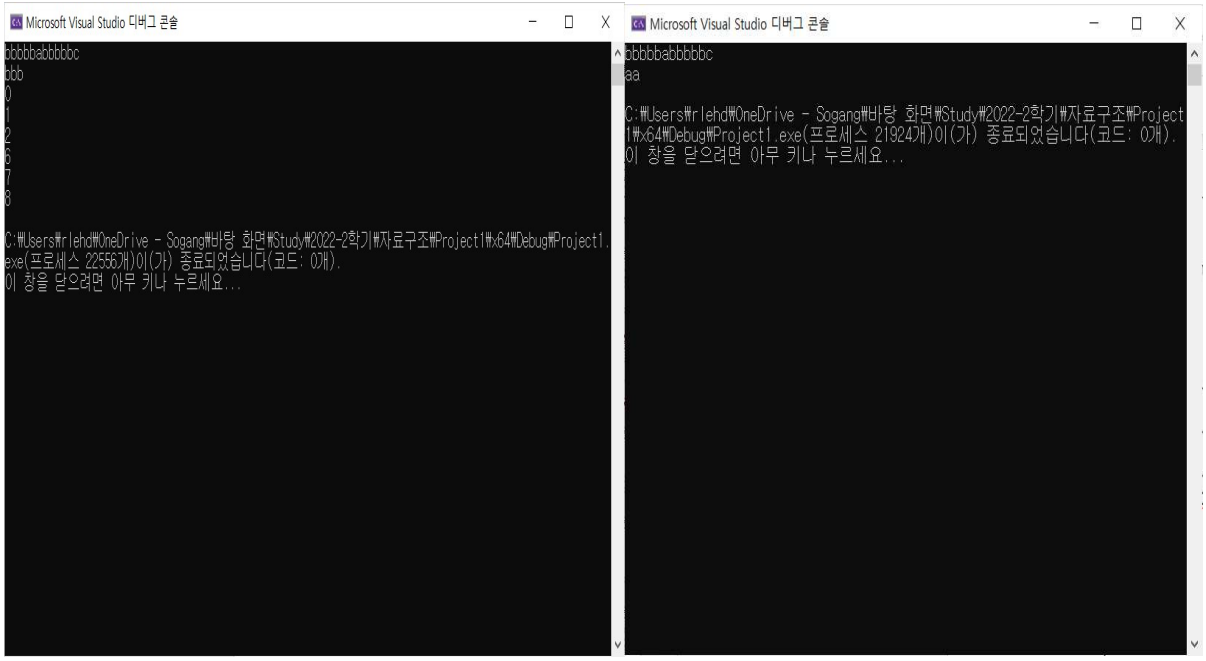
get the string and store it

get the pattern to search for and store it

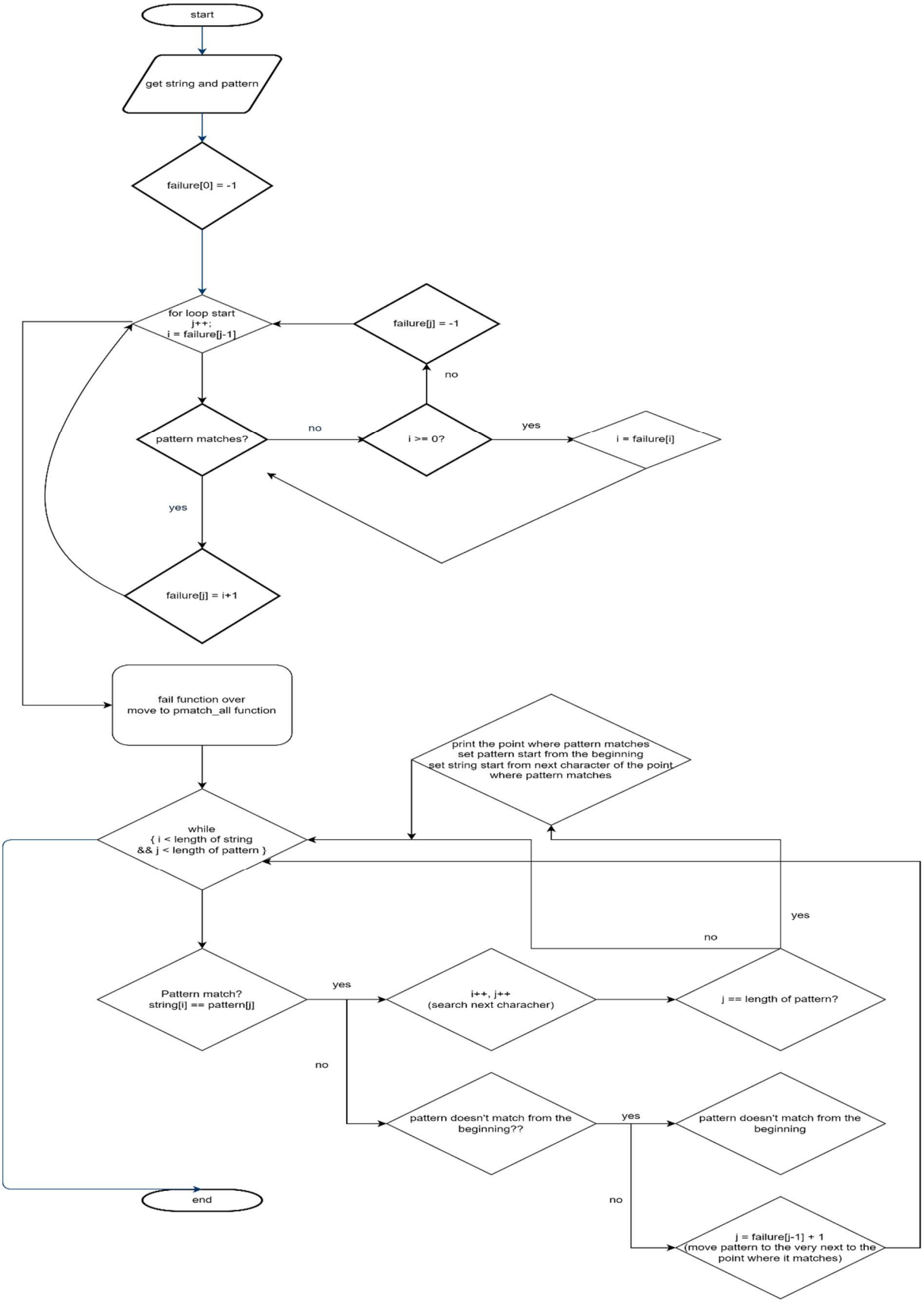
use ‘fail function’ to analysis pattern

use ‘pmatch_all function’ to print out all the points where the pattern matches string

<Test Examples>

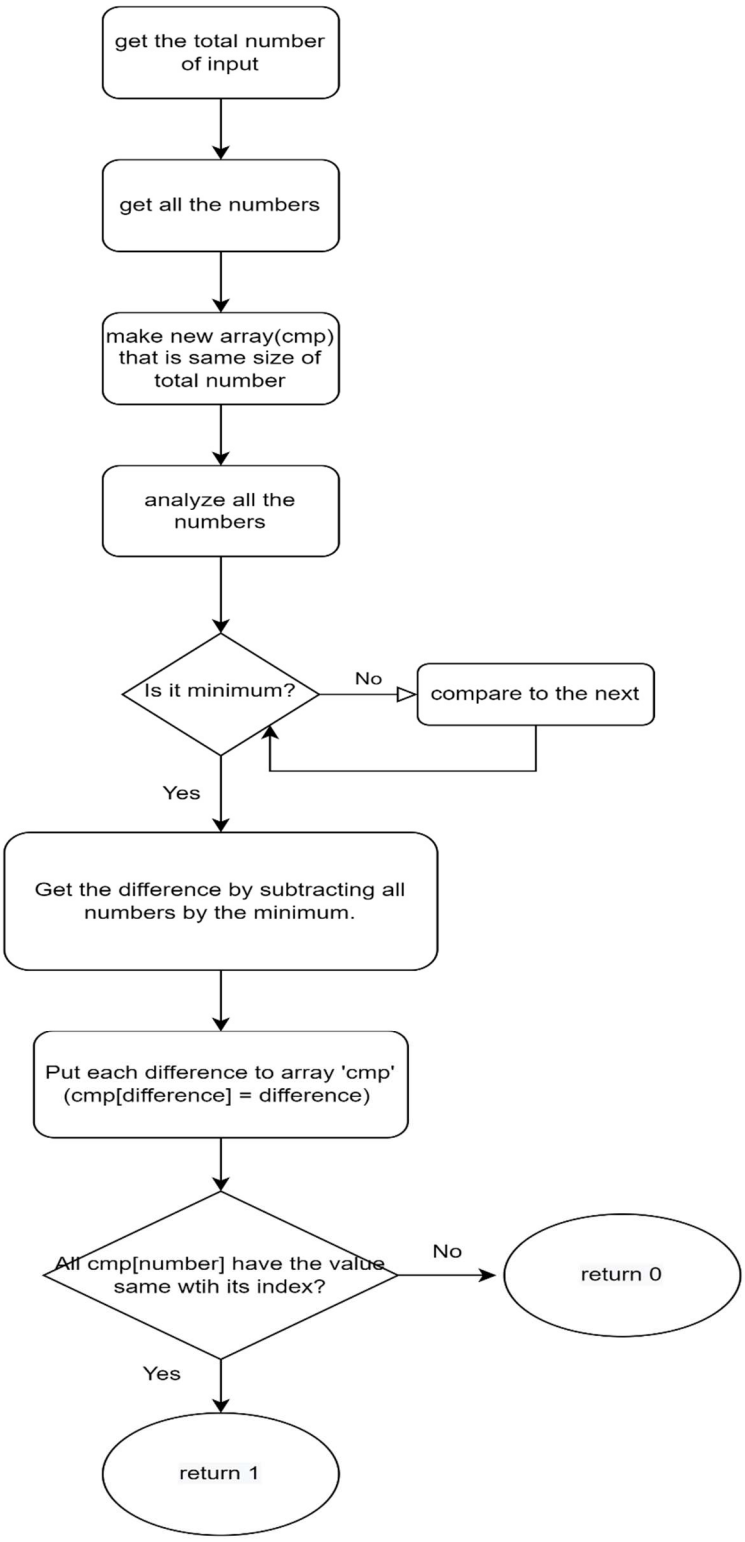


<Flowchart>


start
get string and pattern
failure[0] = -1
for loop start
j++;
i = failure[j-1]
failure[j] = -1
no
pattern matches?
no
i >= 0?
yes
i = failure[i]
yes
failure[j] = i+1
fail function over
move to pmatch_all function
print the point where pattern matches
set pattern start from the beginning
set string start from next character of the point
where pattern matches
while
{ i < length of string
&& j < length of pattern }
yes
no
Pattern match?
string[i] == pattern[j]
yes
i++, j++
(search next characher)
j == length of pattern?
no
pattern doesn't match from the
beginning??
yes
pattern doesn't match from the
beginning
no
j = failure[j-1] + 1
(move pattern to the very next to the
point where it matches)
end

2. check_array() with time complexity $O(n)$

<Flowchart>



<Pseudo Code>

Variable: len(number of inputs), str(set of input numbers), min(minimum number of inputs), cmp(array for storing difference of each number subtracted by minimum)

Function:

1. main function

Get len which means the total number of input numbers

Allocate size of len to str to get the numbers

Get all of numbers

Use ‘check_array’ function and get the result 1 or 0

Print result value

2. int check_array(int len, int* str)

Make ‘cmp’ array same size with str

Get the minimum of numbers in str

Get all the differences by subtracting each number by minimum

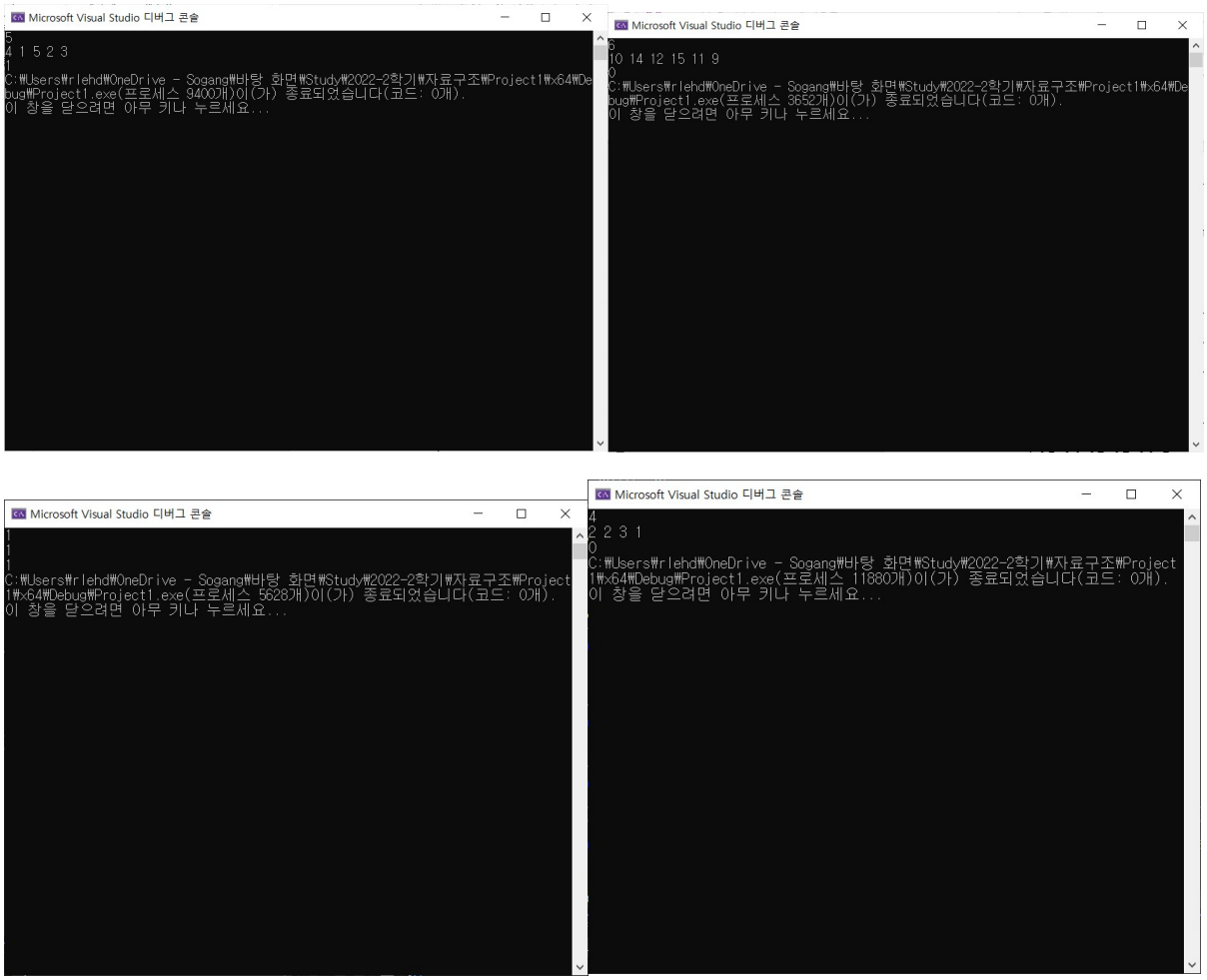
Put each difference to cmp[difference]

Check the array ‘cmp’ whether each index has a corresponding value

If) all indexes have corresponding value, return 1

Else) return 0

<Test Examples>



3. sorting by lexical order

<Pseudo Code>

Variable: fp(file pointer), num(number of input lines), str(2 dimensional array to store each line of string from text file)

Function:

1. Main function

Get the number and dynamically allocate 2-dimensional array named “str”

Open the text file and read each line and store it in “str”

Sorting the lines by lexical order

Print all the lines sorted by lexical order

Free the memory dynamically allocated

2. void sorting(char* str[], int num)

(This sorting function uses same idea of “selection sort”)

Select one string as a ‘standard’ and select the other string to compare

Compare and decide which string is minimum(precedence by lexical order)

Swap the ‘standard string’ with ‘minimum string’

Do this job for all the numbers

3. int string_Compare(char* c1, char* c2)

Get two string to compare

While) both of strings reach to EOF

First compare by “last name”

If) right string is bigger, return -1,

If) two strings are combination of uppercase and lowercase alphabet and lowercase comes later than uppercase in alphabetical order, return 1

Else if) left is bigger, return 1

If) two strings are combination of uppercase and lowercase alphabet and lowercase comes later than uppercase in alphabetical order, return -1

Else) if both have same character, move to next character to compare

If) both strings have different length of last name

If) left is shorter, return -1

Else if) right is shorter, return 1

Else if) same length in last name, go to first name to compare

Do the same thing above in comparing “first name”

After while loop is over without return,

If) left is shorter, return -1

Else if) right is shorter, return 1

` Else) return -2(error)

4. void SWAP(char* c1, char* c2)

Allocate “temp” array to use temporarily to store other string

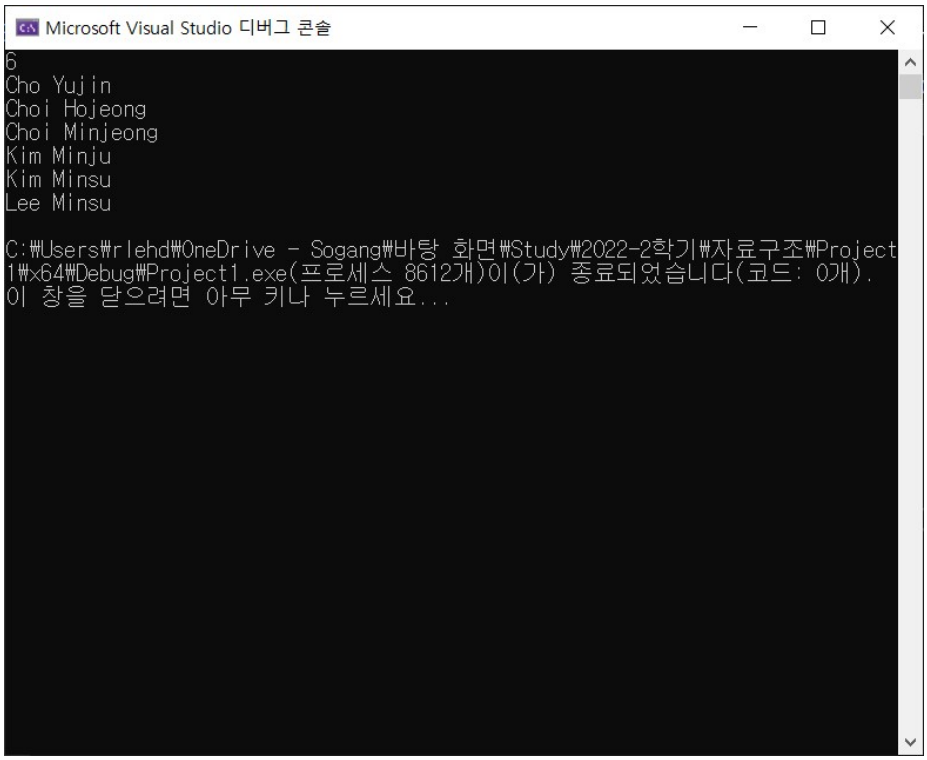
Put ‘c1’ string to temp

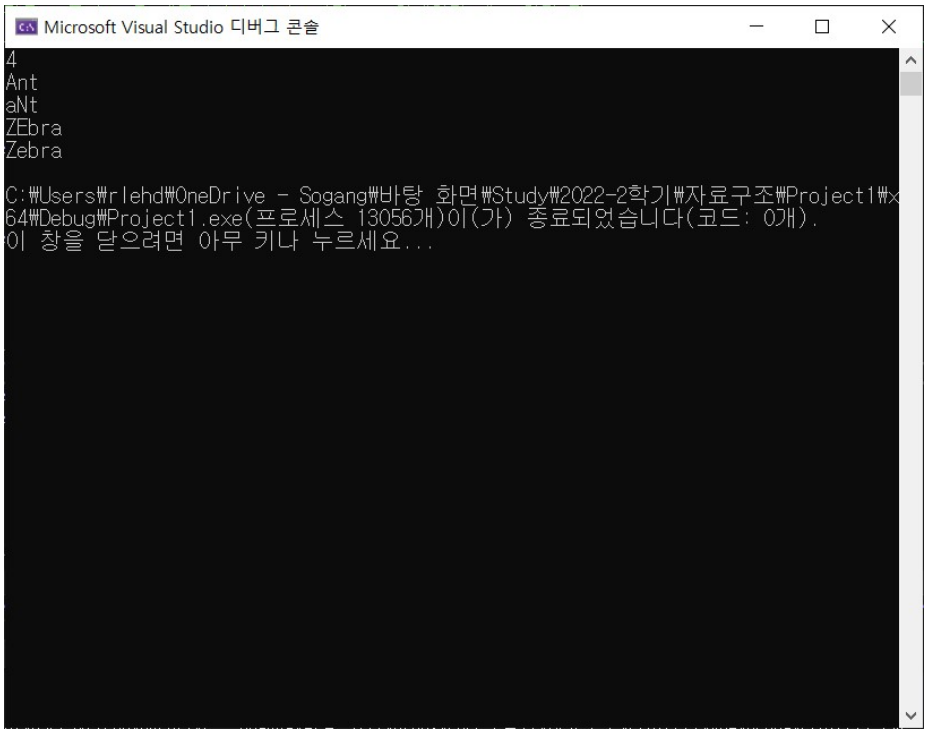
Put ‘c2’ string to ‘c1’ string

Put temp to ‘c1’ string

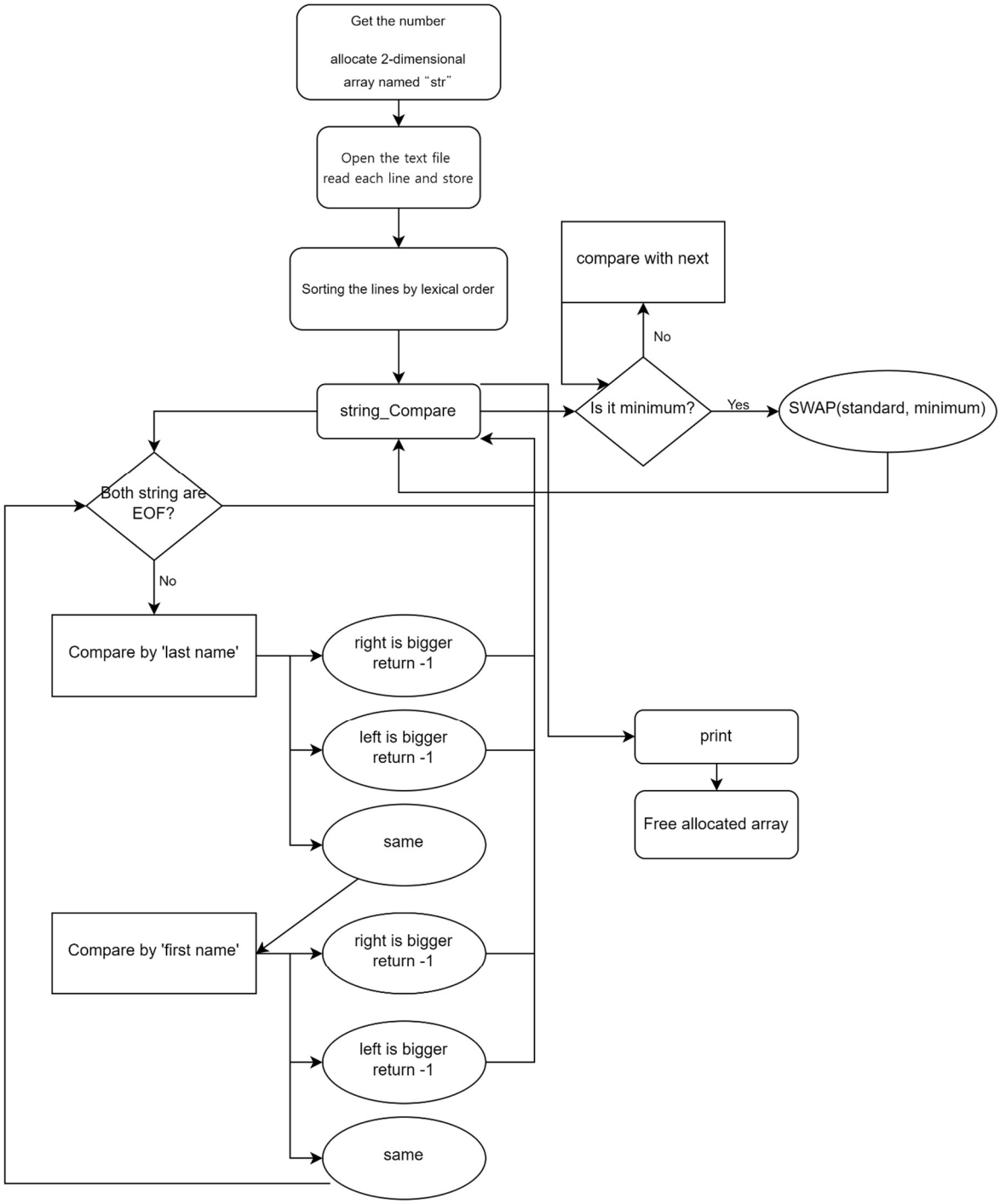
Free “temp” array dynamically allocated

<Test Examples>





<Flowchart>


Get the number
allocate 2-dimensional array named "str"
Open the text file read each line and store
Sorting the lines by lexical order
string_Compare
compare with next
No
Is it minimum?
Yes
SWAP(standard, minimum)
Both string are EOF?
No
Compare by 'last name'
right is bigger return -1
left is bigger return -1
same
print
Free allocated array
Compare by 'first name'
right is bigger return -1
left is bigger return -1
same