Term project

Team 9

201835546 Ham geonwook

202035318 Kim wonjong

202035352 Ahn hyunjin

201834735 Lee jiheon

Software development Process:

**Step 1: understand the problem**

**(1)setup**

read file, store the data (1.array 2.linked list)

file open🡪array {data1, data2, data3}, store data number

file open🡪linked list; head->data1->data2->data3

-cases to consider

no data open, data exceeded(array)

(2) Search for “Choi”

Search from registration data and array structure

If person start with “Choi” found, print all information about him

* Search in the array **P1-1**
* Search in the linked list **P1-2**

Case to consider : no name start with “Choi”, can’t find file to open

(3) Search for all from Gachon University

Search from registration data and array structure

If Gachon University found, print all information about person include Gachon University

* Search in the array **P2-1**
* Search in the linked list **P2-2**

Case to consider : no case of Gachon University, can’t find file to open

(4)Sort the data in the array in tag# order - **P3-1**

data1.tag = 2, data2.tag = 3, data3.tag = 1

{data1, data2, data3} 🡪 sort 🡪 {data3, data1, data2}

-cases to consider

no data, same tag

(5)Create a linked list using the sorted data - **P4-1**

{data3, data1, data2} 🡪 head->data3->data1->data2

-cases to consider

no data

(6) – **P5-1**

Sort the data in the array in age group order(using selection sort) & Write the sorted data to a text file.

Copy array and use bubble sorting, sort the data in the array.

->print at textfile 5-1.txt

Ex) data5.age = 70, data6.age = 39, data7.age = 38

{data5, data6, data7} 🡪 sort 🡪 {data7, data6, data5} 🡪 print at 5-1txt

-cases to consider-

No data, Same age

(7) – **P6-1**

All “Choi”s canceled registration. Remove the data from array. Print result

If data name include Choi, delete data. Save array. Print array.

Ex) {data1, data2, data3} 🡪 data2 include “Choi” 🡪 {data1,data3}

-cases to consider-

Choi is continuous, similar Cho

**- P6-2**

All “Choi”s canceled registration. Remove the data from the linked list. Print result

If data name include Choi, delete node. Save list. Print list.

Ex) {data1, data2, data3} 🡪 data2 include “Choi” 🡪 {data1,data3}

-cases to consider-

Choi is continuous, similar Cho

(8) – **P7-1**

One “Paik” registered late. Add the data to the array

1. Even after adding the data, need to keep the ascending order by age.
2. All affected data will be moved.
3. Print result

Paik's information is as follows.

* 100/2020-11-30/yes/Ildang Paik/22/Gachon University/engineer

ex ) 100/2020-11-30/yes/Ildang Paik/22/Gachon University/engineer 🡨 add data

22/2020-06-29/no/Tongbang Cho/29/Northwestern University/marketer

5/2020-06-12/yes/Chunyong Park/48/University of Cambridge/student

23/2020-06-15/yes/Seungmin Cho/71/Stanford University/professor

- **P7-2**

One “Paik” registered late. Add the data to the linked list

1. Even after adding the data, need to keep the ascending order by tag.
2. Print result

Paik's information is as follows.

* 100/2020-11-30/yes/Ildang Paik/22/Gachon University/engineer

ex ) 5/2020-06-12/yes/Chunyong Park/48/University of Cambridge/student

22/2020-06-29/no/Tongbang Cho/29/Northwestern University/marketer

23/2020-06-15/yes/Seungmin Cho/71/Stanford University/professor

100/2020-11-30/yes/Ildang Paik/22/Gachon University/engineer 🡨 add data

(9) **P8-1**

Copy the names of most recent data in the array for transmission to a remote computer.

1. Last 5 data in array(name)
2. Compute original data checksum (using bitwise ExclusiveOR)
3. Attach it to the copy.
4. Compute the checksum in the copied data
5. Compare it against the checksum in the original data.
6. Confirm that the two data are the same.

ex ) [original data] [copied data] [result]

checksum 55 55 Same(55=55)

name Choi Choi

Kim Kim

… …

**Step2: outline a solution**

**(1)setup**

Fileopen

loop for file open

open one line and store the array

return array

loop for file open

open one line and store the linked list

return head

(2) Search for “Choi”

Read the data array - **P1-1**

Char name in registration data txt file

Loop for find “Choi”

Open file in reading mode

Search name “Choi”

If no file

Print can’t open file

If find “Choi”

Get including data

Print all data include “Choi”

Close file

Read the data linked list head - **P1-2**

Loop for find “Choi”

Set head

If find “Choi”

Print all data including name “Choi”

Print linked list

(3) Search for all from Gachon University

Read the data array – **P2-1**

Char name in registration data txt file

Loop for find Gachon University

Open file in reading mode

Search name Gachon University

If no file

Print can’t open file

If find Gachon University

Get including data

Print all data include Gachon University

Close file

Read the data linked list head – **P2-2**

Loop for find Gachon University

Set head

If find Gachon University

Print all data of including Gachon University

Print linked list

(4)Sort the data in the array in tag# order - **P3-1**

Read the data array

Loop for Sort

Sort the data

(5)Create a linked list using the sorted data - **P4-1**

Read the data array

Loop to make a linked list

Data.next = next data

Head.next = first data

Return head

(6) – **P5-1**

Read the data array

Loop for bubble sort

Sort the order by age order

Return array

Loop for write file

Print array at textfile 5-1.txt

Return head

(7) – **P6-1**

Read the data array

Loop for find “Choi”

If find {Move to next data}

Else {save at temp}

Copy temp to array

Return data\_num

Print array

- **P6-2**

Read the data linked list head

Loop for find “Choi”

Set head

If find {delete node}

Else {move to next node}

Return head

Print linked list

(8) – **P7-1**

Read the data array

Add new data at the end of the array

Loop for bubble sort

Sort ascending by age

Print array

- **P7-2**

Read the data linked list

Loop for linked list

Add a new node to the end of the linked list

Loop for bubble sort

Sort ascending by tag

Print linked list

(9) – **P8-1**

Read data array

Loop for array

Find data of last 5 names in array

Copy to new file

Compute original data checksum

attach copied file

Read copied file

Loop for file read

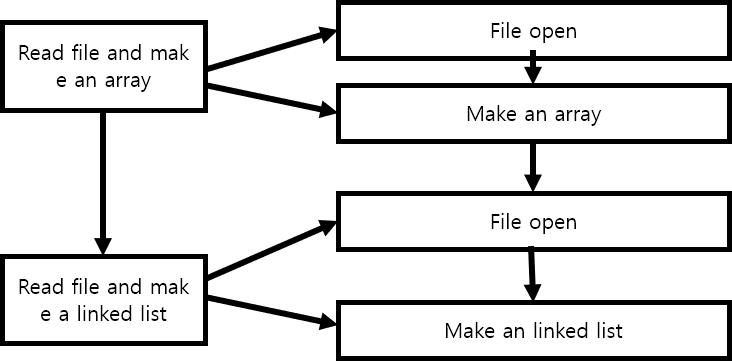
Compute the checksum in the copied data

Compare the original checksum and copied data checksum

Print compare result

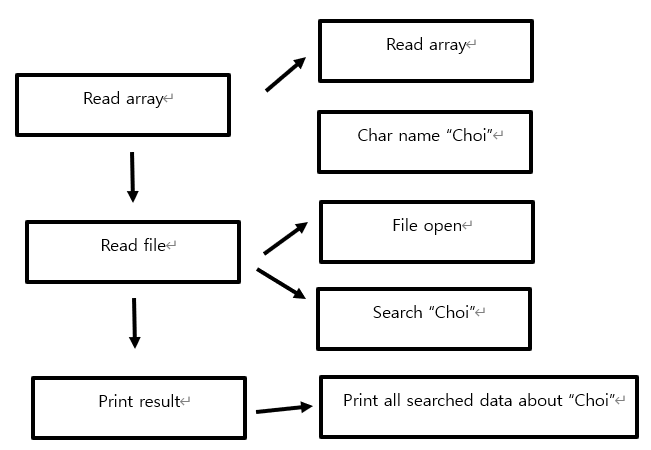
**Step 3: form a program structure**

(**1)setup**

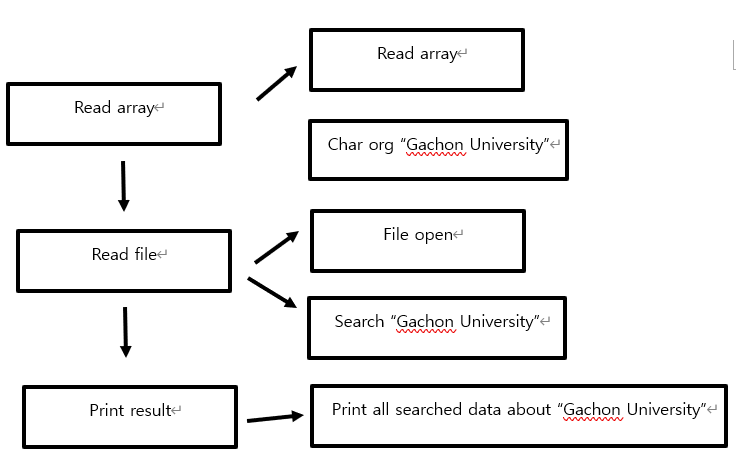


(2) Search for “Choi”

**P1-1**

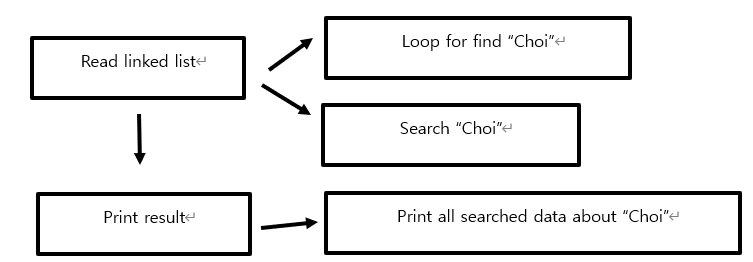
****

**P1-2**

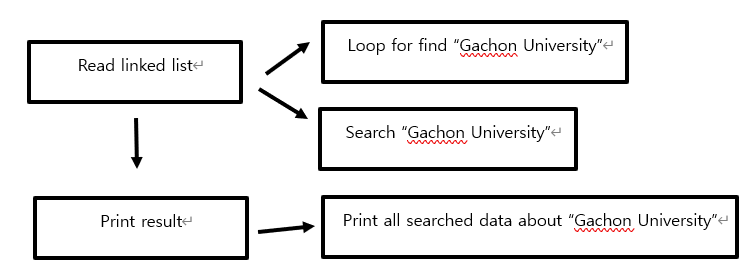


(3) Search for all from Gachon University

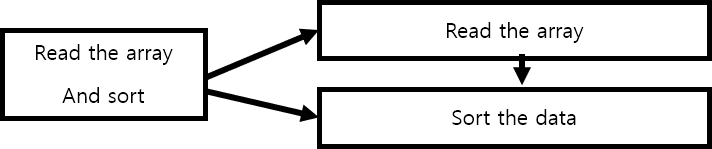
**P2-1**

****

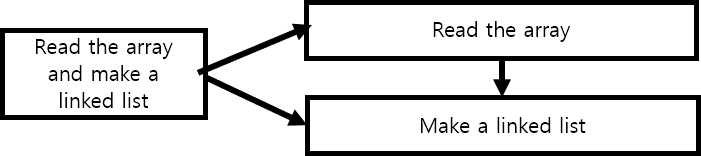
**P2-2**



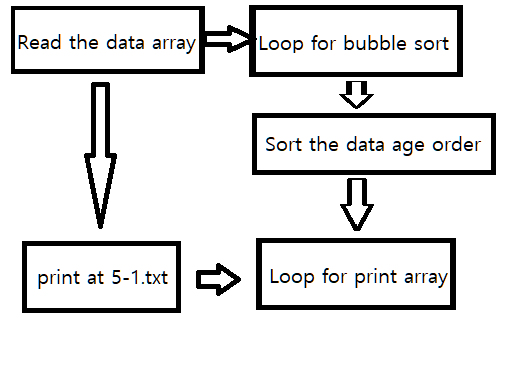
(4)Sort the data in the array in tag# order **- P3-1**



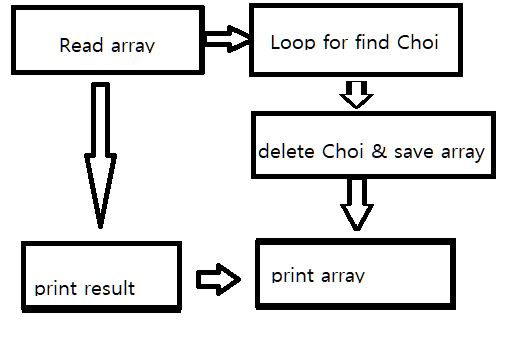
(5)Create a linked list using the sorted data **- P4-1**



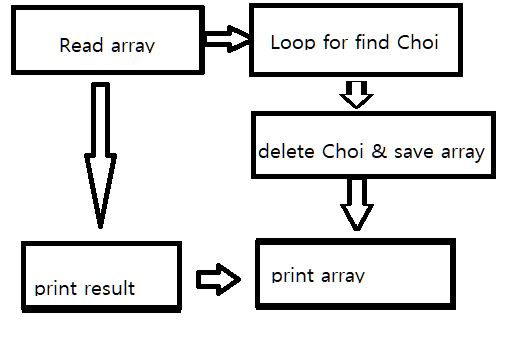
(6) – **P5-1**



(7) – **P6-1**



- **P6-2**



(8) – **P7-1**

Add new data

at the end of the array

Read the

data array

Sort ascending by age

Print data in array

Print result

- **P7-2**

Add new node

to the end of the linked list

Read the

data linked list

Sort ascending by tag

Print data linked list

Print result

(9) – **P8-1**

Find data of last

5 names In array

Read the

data array

Copy to new file

Compute original data checksum

And attach copied file

Compute the original and

Copied data checksum

Compute the checksum in the

copied data

Print result

Compare the original checksum and copied data checksum

**Step 4: write a program outline (pseudo code)**

**(1)setup**

array\_open(array)

{

file = fopen("registraion\_data.txt", "r");

while (fscanf(regi\_file, array))

i++;

data\_number = i;

fclose(regi\_file);

return data\_number;

}

Linked\_list\_open()

{

head;

prev\_node = head;

regi\_file = fopen("registraion\_data.txt", "r");

node;

while (fscanf(regi\_file, node)

{

prev\_node->next = node;

prev\_node = node;

node = new\_node;

}

prev\_node->next = NULL; //last node

}

fclose(regi\_file);

return head;

}

(2) Search for “Choi”

**P1-1**

Array\_findname(Data \*regi\_data){

FILE\* myfile;

Char name=”Choi”;

Char array;

Char fname=”registration\_data,txt”;

myfile = fopen(fname, "r");

if (no file){

printf(“Can’t find %s”, fname);}

for (int i=0; i<30; i++){

fgets(buffer,300,myfile);

if (find string name){

printf(“All searched data in array”);}

}

fclose(myfile);

}

**P1-2**

Search\_Choi(Data \*head){

Data \*node=head->next, \*prev=head;

While(node){

If((node->name Choi) != NULL){

Printf(All searched data in linked list);}

Prev=node;

Node=node->next;

}

}

(3) Search for all from Gachon University

**P2-1**

Array\_findorg (Data \*regi\_data){

FILE\* myfile;

Char org=”Gachon University”;

Char array;

Char fname=”registration\_data,txt”;

myfile = fopen(fname, "r");

if (no file){

printf(“Can’t find %s”, fname);}

for (int i=0; i<30; i++){

fgets(buffer,300,myfile);

if (find string org){

printf(“All searched data in array”);}

}

fclose(myfile);

}

**P2-2**

Search\_Gachon(Data \*head){

Data \*node=head->next, \*prev=head;

While(node){

If((node->org Gachon University) != NULL){

Printf(All searched data in linked list);}

Prev=node;

Node=node->next;

}

}

(4)Sort the data in the array in tag# order **- P3-1**

sort\_tag(array, sorted\_array, data\_number)

{

for (int i = 0; i < data\_number; i++)

sorted\_array[i] = array[i];

for (int i = 0; i < data\_number - 1; i++)

{

for (int j = 0; j < data\_num - 1 - i; j++)

{

if (sorted\_array [j].tag > sorted\_array [j + 1].tag)

{

temp = sorted\_array [j];

sorted\_array [j] = sorted\_array [j + 1];

sorted\_array [j + 1] = temp;

}

}

}

}

(5)Create a linked list using the sorted data **- P4-1**

Make\_linked\_list(sorted\_array, data\_number)

{

sorted\_head;

node, prev\_node;

node = sorted\_array[0];

sorted\_head ->next = node; prev\_node = node;

for (int i = 1; i < data\_number; i++)

{

node = new\_node; node = sorted\_array[i];

prev\_node->next = node; prev\_node = node;

}

node->next = NULL; //last node

return sorted\_head;

}

(6) – **P5-1**

**//bring array and data\_num**

ageOrder(array, data\_num)

{

**//sort data age order with bubble sort**

array temp;

for loop i

{ for loop j

{

if (array[j].age > array[j + 1].age)

{

temp = array[j];

array[j] = array[j + 1];

array[j + 1] = temp;

}

}

}

for loop

{

temp = array[k + 1];

array[k] = temp;

}

**//write data to 5-1.txt**

age\_file = fopen("5-1.txt", "w");

{

for loop

{

fprintf array at 5-1.txt

}

} fclose(age\_file);

}

(7) – **P6-1**

int choi(Data\* array, int data\_num)

{

Data temp[100];

int i=0, j=0;

for loop //Loop for find Choi

{

char\* ptr = strstr(array[i].name,"Choi");

if (ptr!= NULL)

{

i++;

if (strstr(array[i].name, "Choi" )!= NULL) // continue Choi

{

i++;

temp[j] = array[i];

j++;

}

else // save at temp

{

temp[j] = array[i];

j++;

}

}

else // save at temp

{

temp[j] = array[i];

j++;

}

}

data\_num = j; // revise data\_num

for loop // copy temp to array

{

array[j] = temp[j];

}

return data\_num;

}

- **P6-2**

Data\* choiNode(Data\* head)

{

**//set ptr & prev**

Data\* ptr = head->next, \* prev = head,\*temp;

while (ptr) {

if (strstr(ptr->name, "Choi") != NULL) **// Find Choi**

{

if (strstr((ptr->next)->name, "Choi") != NULL) **// if next node is "Choi" again**

{

prev->next = ((ptr->next)->next); **//delete two node**

prev = ptr;

ptr = ptr->next;

}

else

{

prev->next = ptr->next; **/\* node deleted \*/**

prev = ptr;

ptr = ptr->next;

}

}

else

{

prev = ptr;

ptr = ptr->next;

}

}

return head;

}

(8) – **P7-1**

void add\_data\_to\_array(Data \*regi\_data, int \*data\_num, int tag, char \*date, char \*fee\_paid, char \*name, int age, char \*organization, char \*job)

{

int new\_data\_num = \*data\_num;

regi\_data[new\_data\_num].tag = tag;

strcpy(regi\_data[new\_data\_num].date, date);

strcpy(regi\_data[new\_data\_num].fee\_paid, fee\_paid);

strcpy(regi\_data[new\_data\_num].name, name);

regi\_data[new\_data\_num].age = age;

strcpy(regi\_data[new\_data\_num].organization, organization);

strcpy(regi\_data[new\_data\_num].job, job);

\*data\_num += 1;

new\_data\_num = \*data\_num;

Data temp;

*// Bubble sort by age*

*for* (int i = 0; i < new\_data\_num; i++)

{

*for* (int j = 0; j < new\_data\_num - 1; j++)

{

*if* (regi\_data[j].age > regi\_data[j + 1].age)

{

temp = regi\_data[j];

regi\_data[j] = regi\_data[j + 1];

regi\_data[j + 1] = temp;

}

}

}

/\* Print array \*/

}

- **P7-2**

void add\_data\_to\_linked(Data \*regi\_head, int tag, char \*date, char \*fee\_paid, char \*name, int age, char \*organization, char \*job)

{

*// Create node to be added*

Data \*new\_node = (Data \*)malloc(sizeof(Data));

new\_node->tag = tag;

strcpy(new\_node->date, date);

strcpy(new\_node->fee\_paid, fee\_paid);

strcpy(new\_node->name, name);

new\_node->age = age;

strcpy(new\_node->organization, organization);

strcpy(new\_node->job, job);

new\_node->next = NULL;

Data \*ptr, \*phead, \*temp;

*// Insert new node into last node*

ptr = regi\_head->next;

*while* (ptr->next != NULL)

{

ptr = ptr->next;

}

ptr->next = new\_node;

*// Bubble sort by tag*

phead = regi\_head;

ptr = phead;

*while* (ptr->next != NULL)

{

*if* (ptr->tag > ptr->next->tag)

{

temp = ptr->next;

ptr->next = ptr->next->next;

temp->next = phead;

phead = ptr = temp;

*continue*;

}

ptr = ptr->next;

}

*// Print added data and sorted linked list*

}

(9) – **P8-1**

void copy\_is\_vaild(Data \*regi\_data, int data\_num)

{

FILE \*copyFile = fopen("8-1.txt", "w");

char checksum;

*// Create checksum insert location on first line.*

fprintf(copyFile, "%s\n", "00");

*// 5 most recent data names in array*

*for* (int i = data\_num - 1; i > data\_num - 6; i--)

{

fprintf(copyFile, "%s\n", regi\_data[i].name);

int j = 0;

*while* (regi\_data[i].name[j] != '\0')

{

*// using bitwise ExclusiveOR*

checksum ^= regi\_data[i].name[j];

j++;

}

}

*// Move the file pointer position to the start point. And Overwrite checksum value.*

fseek(copyFile, 0, SEEK\_SET);

fprintf(copyFile, "%d\n", checksum);

fclose(copyFile);

compare\_checksum();

}

void compare\_checksum()

{

FILE \*readFile = fopen("8-1.txt", "r");

char str[25];

char read\_checksum = '\0', calculate\_checksum = '\0';

int line = 1;

*while* (fgets(str, 25, readFile) != NULL)

{

*// Since there is a checksum on the first line, it counts by excluding the first line.*

*if* (line != 1)

{

int i = 0;

*// Currently, str contains '\n', so exclude it.*

*while* (str[i] != '\n')

{

*// using bitwise ExclusiveOR*

calculate\_checksum ^= str[i];

i++;

}

}

*else*

{

int temp = atoi(str);

read\_checksum += temp;

}

line++;

}

fclose(readFile);

*if* (read\_checksum == calculate\_checksum)

{

/\* Print Same \*/

}

*else*

{

/\* Print not equal \*/

}

}

Source code :

/\*

setup code(array\_open, print\_array, linked\_open, print\_linked) - 202035318 Kim wonjong

\*/

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <string.h>

#include <stdlib.h>

//#define DATA\_NUM 30

typedef struct DATA

{

int tag;

char date[20];

char fee\_paid[10];

char name[25];

int age;

char organization[30];

char job[15];

struct DATA\* next;

} Data;

//-----------------------------

//Function prototype

//-----------------------------

int array\_open(Data\*);

void print\_array(Data\*, int); // setup

Data\* linked\_open();

void print\_linked(Data\*);

void array\_findname(Data\*); //1-1

void array\_findorg(Data\*); //2-1

Data\* search\_Choi(Data\*); //1-2

Data\* search\_Gachon(Data\*); //2-2

void sort\_tag(Data\*, Data\*, int);

Data\* tag\_linked(Data\*, int);

void ageOrder(Data\*, int);

int choi(Data\*, int);

Data\* choiNode(Data\*);

void add\_data\_to\_array(Data\*, int\*, int, char\*, char\*, char\*, int, char\*, char\*);

void add\_data\_to\_linked(Data\*, int, char\*, char\*, char\*, int, char\*, char\*);

void copy\_is\_vaild(Data\*, int);

void compare\_checksum();

//-----------------------------

//MAIN

//-----------------------------

int main()

{

Data regi\_data[50];

Data\* regi\_head;

int data\_num;

Data tsort\_data[50]; //tsort\_data -> sorted data in tag order

Data\* tsort\_head; //tsort linked list's head

data\_num = array\_open(regi\_data);

print\_array(regi\_data, data\_num);

printf("\n");

regi\_head = linked\_open();

print\_linked(regi\_head->next);

//P1-1, 202035352 안현진

printf("\n===P1-1===\n");

array\_findname(regi\_data);

//P1-2, 202035352 안현진

printf("\n===P1-2===\n");

search\_Choi(regi\_head);

//P2-1, 202035352 안현진

printf("\n===P2-1===\n");

array\_findorg(regi\_data);

//P2-2, 202035352 안현진

printf("\n===P2-2===\n");

search\_Gachon(regi\_head);

//P3-1, 202035318 Kim wonjong

printf("\n===P3-1===\n");

sort\_tag(regi\_data, tsort\_data, data\_num);

print\_array(tsort\_data, data\_num);

//P4-1, 202035318 Kim wonjong

printf("\n===P4-1===\n");

tsort\_head = tag\_linked(tsort\_data, data\_num);

print\_linked(tsort\_head->next);

//P5-1, 201835546 함건욱

ageOrder(regi\_data, data\_num);

//P6-1, 201835546 함건욱

printf("\n===P6-1===\n");

data\_num = choi(regi\_data, data\_num);

printf("Choi canceled registration\n");

print\_array(regi\_data, data\_num);

//P6-2, 201835546 함건욱

printf("\n===P6-2===\n");

regi\_head = choiNode(regi\_head);

printf("Choi cencled list\n");

print\_linked(regi\_head->next);

// P7-1, 201834735 LeeJiheon

add\_data\_to\_array(regi\_data, &data\_num, 100, "2020-11-30", "yes", "Ildang Paik", 22, "Gachon University", "engineer");

// P7-2, 201834735 LeeJiheon

add\_data\_to\_linked(regi\_head, 100, "2020-11-30", "yes", "Ildang Paik", 22, "Gachon University", "engineer");

// P8-1, 201834735 LeeJiheon

copy\_is\_vaild(regi\_data, data\_num);

return 0;

}

//-----------------------------

//Read registraion\_data.txt and store the data in a struct array

//-----------------------------

int array\_open(Data\* regi\_data)

{

int i = 0, data\_num = 0;

FILE\* regi\_file;

regi\_file = fopen("registraion\_data.txt", "r");

if (regi\_file == NULL)

printf("file open error\n");

else

{

while (fscanf(regi\_file, "%d %\*[/] %[^/] %\*[/] %[^/] %\*[/] %[^/] %\*[/] %d %\*[/] %[^/] %\*[/] %s", &regi\_data[i].tag, regi\_data[i].date, regi\_data[i].fee\_paid, regi\_data[i].name, &regi\_data[i].age, regi\_data[i].organization, regi\_data[i].job) == 7)

i++;

}

data\_num = i;

fclose(regi\_file);

return data\_num;

}

//-----------------------------

//Print array

//-----------------------------

void print\_array(Data\* regi\_data, int data\_num)

{

printf("===array===\n");

for (int i = 0; i < data\_num; i++)

printf("%d %s %s %s %d %s %s\n", regi\_data[i].tag, regi\_data[i].date, regi\_data[i].fee\_paid, regi\_data[i].name, regi\_data[i].age, regi\_data[i].organization, regi\_data[i].job);

printf("======\n");

}

//-----------------------------

//Store the data in a linked list

//-----------------------------

Data\* linked\_open()

{

int i = 0;

FILE\* regi\_file;

Data\* node;

Data\* head;

Data\* prev\_node;

head = (Data\*)malloc(sizeof(Data));

prev\_node = head;

regi\_file = fopen("registraion\_data.txt", "r");

if (regi\_file == NULL)

printf("file open error\n");

else

{

node = (Data\*)malloc(sizeof(Data));

while (fscanf(regi\_file, "%d %\*[/] %[^/] %\*[/] %[^/] %\*[/] %[^/] %\*[/] %d %\*[/] %[^/] %\*[/] %s", &node->tag, node->date, node->fee\_paid, node->name, &node->age, node->organization, node->job) == 7)

{

prev\_node->next = node;

prev\_node = node;

node = (Data\*)malloc(sizeof(Data));

}

free(node); //free trash node

prev\_node->next = NULL; //last node

}

fclose(regi\_file);

return head;

}

//-----------------------------

//Print linked list

//-----------------------------

void print\_linked(Data\* ptr)

{

printf("===linked list===\n");

while (ptr != NULL)

{

printf("%d %s %s %s %d %s %s\n", ptr->tag, ptr->date, ptr->fee\_paid, ptr->name, ptr->age, ptr->organization, ptr->job);

ptr = ptr->next;

}

printf("======\n");

}

//-----------------------------

//P1-1 / 202035352 안현진

// Search for "Choi" in the array

// if found, print all info of the persons

//-----------------------------

void array\_findname(Data\* regi\_data) {

FILE\* myfile;

char name[20] = "Choi";

char buffer[300];

char buffer\_2[300];

int line\_num = 0;

int count = 29;

char org[35];

char fname[30] = "registraion\_data.txt";

printf("----------------search array---------------\n");

printf("name to search: Choi");

myfile = fopen(fname, "r");

if (myfile == NULL) {

fprintf(stderr, "Can't open the file%s \n", fname);

exit(1);

}

printf("\n");

for (int i = 0; i < 30; i++) {

fgets(buffer, 300, myfile);

if (strstr(buffer, name))//function to find string

{

printf("%d %s %s %s %d %s %s\n", regi\_data[i].tag, regi\_data[i].date, regi\_data[i].fee\_paid, regi\_data[i].name, regi\_data[i].age, regi\_data[i].organization, regi\_data[i].job);

}

}

fclose(myfile);

}//FINISH

//-----------------------------

//P1-2 / 202035352 안현진

// Search for "Choi" in the linked list

// if found, print all info of the persons

//-----------------------------

Data\* search\_Choi(Data\* head)

{

Data\* node = head->next, \* prev = head; //set node & prev

while (node)

{

if (strstr(node->name, "Choi") != NULL)

{

printf("%d %s %s %s %d %s %s\n", node->tag, node->date, node->fee\_paid, node->name, node->age, node->organization, node->job);

}

prev = node;

node = node->next;

}

}//finish

//-----------------------------

//P2-1 / 202035352 안현진

// Search for all from Gachon University in the array

// if found, print all info of the persons

//-----------------------------

void array\_findorg(Data\* regi\_data) {

FILE\* myfile;

char name[20];

char buffer[300];

char buffer\_2[300];

int line\_num = 0;

int count = 29;

char org[35] = "Gachon University";

char fname[30] = "registraion\_data.txt";

printf("----------------search array---------------\n");

printf("org to search: Gachon University");

myfile = fopen(fname, "r");

if (myfile == NULL) {

fprintf(stderr, "Can't open the file%s \n", fname);

exit(1);

}

printf("\n");

for (int i = 0; i < 30; i++) {

fgets(buffer\_2, 300, myfile);

if (strstr(buffer\_2, org))//function to find string

{

printf("%d %s %s %s %d %s %s\n", regi\_data[i].tag, regi\_data[i].date, regi\_data[i].fee\_paid, regi\_data[i].name, regi\_data[i].age, regi\_data[i].organization, regi\_data[i].job);

}

}

fclose(myfile);

}//FINISH

//-----------------------------

//P2-2 / 202035352 안현진

// Search for all from Gachon University in the linked list

// if found, print all info of the persons

//-----------------------------

Data\* search\_Gachon(Data\* head)

{

Data\* node = head->next, \* prev = head; //set node & prev

while (node)

{

if (strstr(node->organization, "Gachon University") != NULL)

{

printf("%d %s %s %s %d %s %s\n", node->tag, node->date, node->fee\_paid, node->name, node->age, node->organization, node->job);

}

prev = node;

node = node->next;

}

}//finish

//-----------------------------

//P3-1 / 202035318 - Kim wonjong

//Sort the data in the array in tag# order

//-----------------------------

void sort\_tag(Data\* regi\_data, Data\* tsort\_data, int data\_num)

{

Data temp;

//tsort initialization

for (int i = 0; i < data\_num; i++)

tsort\_data[i] = regi\_data[i];

//Sort the data

for (int i = 0; i < data\_num - 1; i++)

{

for (int j = 0; j < data\_num - 1 - i; j++)

{

if (tsort\_data[j].tag > tsort\_data[j + 1].tag)

{

temp = tsort\_data[j];

tsort\_data[j] = tsort\_data[j + 1];

tsort\_data[j + 1] = temp;

}

}

}

}

//-----------------------------

//P4-1 / 202035318 - Kim wonjong

//Create a linked list using the sorted data

//-----------------------------

Data\* tag\_linked(Data\* tsort\_data, int data\_num)

{

Data\* tsort\_head;

Data\* node, \* prev\_node;

//node and head allocation

tsort\_head = (Data\*)malloc(sizeof(Data));

node = (Data\*)malloc(sizeof(Data));

\*node = tsort\_data[0];

tsort\_head->next = node;

prev\_node = node;

//list -> linked list

for (int i = 1; i < data\_num; i++)

{

node = (Data\*)malloc(sizeof(Data));

\*node = tsort\_data[i];

prev\_node->next = node;

prev\_node = node;

}

node->next = NULL; //last node

return tsort\_head;

}

//------------------------------------------

//P5-1 / 201835546 함건욱

//Sort the data in the array in age group order (using selection sort – self-study)

//\*\*“age group” means 10, 20, 30, ….

//Write the sorted data to a text file. (Mark P5 - 1 in code, submit textfile 5 - 1.txt)

//------------------------------------------

void ageOrder(Data\* array, int data\_num) //bring array and data\_num

{

Data temp;

for (int i = 0; i < data\_num; i++) //sort data age order

{

for (int j = 0; j < data\_num - i; j++)

{

if (array[j].age > array[j + 1].age)

{

temp = array[j];

array[j] = array[j + 1];

array[j + 1] = temp;

}

}

}

for (int k = 0; k < data\_num; k++)

{

temp = array[k + 1];

array[k] = temp;

}

FILE\* age\_file; // FIle header

age\_file = fopen("5-1.txt", "w"); //write data to 5-1.txt

{

fprintf(age\_file, "===age order array===\n");

for (int i = 0; i < data\_num; i++)

{

fprintf(age\_file, "%d %s %s %s %d %s %s\n", array[i].tag, array[i].date, array[i].fee\_paid, array[i].name, array[i].age, array[i].organization, array[i].job);

}

fprintf(age\_file, "======\n");

}

fclose(age\_file);

}

//------------------------------------------

//P6-1 / 201835546 함건욱

//All “Choi”s canceled registration. Remove the data from the array

//------------------------------------------

int choi(Data\* array, int data\_num)

{

Data temp[100];

int i = 0, j = 0;

for (i = 0; i < data\_num; i++)

{

char\* ptr = strstr(array[i].name, "Choi");

if (ptr != NULL)

{

i++;

if (strstr(array[i].name, "Choi") != NULL)

{

i++;

temp[j] = array[i];

j++;

}

else

{

temp[j] = array[i];

j++;

}

}

else

{

temp[j] = array[i];

j++;

}

}

data\_num = j;

for (j = 0; j < data\_num; j++)

{

array[j] = temp[j];

}

return data\_num;

}

//------------------------------------------

//P6-2 / 201835546 함건욱

//All “Choi”s canceled registration. Remove the data from the linked list

//------------------------------------------

Data\* choiNode(Data\* head)

{

Data\* ptr = head->next, \* prev = head, \* temp; //set ptr & prev

while (ptr)

{

if (strstr(ptr->name, "Choi") != NULL)

{

if (strstr((ptr->next)->name, "Choi") != NULL) // if nex next node is "Choi" again

{

prev->next = ((ptr->next)->next); //delete two node

prev = ptr;

ptr = ptr->next;

}

else

{

prev->next = ptr->next; /\* node deleted \*/

prev = ptr;

ptr = ptr->next;

}

}

else

{

prev = ptr;

ptr = ptr->next;

}

}

prev = head; //sort tag order

ptr = prev;

while (ptr->next != NULL)

{

if (ptr->tag > ptr->next->tag)

{

temp = ptr->next;

ptr->next = ptr->next->next;

temp->next = prev;

prev = ptr = temp;

continue;

}

ptr = ptr->next;

}

return head;

}

// ------------------------------------------

// P7-1 / 201834735 LeeJiheon

// One “Paik” registered late. Add the data to the array(sorted order)

// ------------------------------------------

void add\_data\_to\_array(Data\* regi\_data, int\* data\_num, int tag, char\* date, char\* fee\_paid, char\* name, int age, char\* organization, char\* job)

{

printf("\n===P7-1===\n");

// Arrays start at 0, so 1 is not added

int new\_data\_num = \*data\_num;

regi\_data[new\_data\_num].tag = tag;

strcpy(regi\_data[new\_data\_num].date, date);

strcpy(regi\_data[new\_data\_num].fee\_paid, fee\_paid);

strcpy(regi\_data[new\_data\_num].name, name);

regi\_data[new\_data\_num].age = age;

strcpy(regi\_data[new\_data\_num].organization, organization);

strcpy(regi\_data[new\_data\_num].job, job);

\*data\_num += 1;

new\_data\_num = \*data\_num;

Data temp;

// Bubble sort by age

for (int i = 0; i < new\_data\_num; i++)

{

for (int j = 0; j < new\_data\_num - 1; j++)

{

if (regi\_data[j].age > regi\_data[j + 1].age)

{

temp = regi\_data[j];

regi\_data[j] = regi\_data[j + 1];

regi\_data[j + 1] = temp;

}

}

}

for (int i = 0; i < new\_data\_num; i++)

{

printf("%d %s %s %s %d %s %s", regi\_data[i].tag, regi\_data[i].date, regi\_data[i].fee\_paid, regi\_data[i].name, regi\_data[i].age, regi\_data[i].organization, regi\_data[i].job);

// Data mark added

if (!strcmp(regi\_data[i].name, name) && regi\_data[i].tag == tag)

{

printf(" <===== added data");

}

printf("\n");

}

printf("==========\n");

}

// ------------------------------------------

// P7-2 / 201834735 LeeJiheon

// One “Paik” registered late. Add the data to the linked list(sorted order)

// ------------------------------------------

void add\_data\_to\_linked(Data\* regi\_head, int tag, char\* date, char\* fee\_paid, char\* name, int age, char\* organization, char\* job)

{

printf("\n===P7-2===\n");

// Create node to be added

Data\* new\_node = (Data\*)malloc(sizeof(Data));

new\_node->tag = tag;

strcpy(new\_node->date, date);

strcpy(new\_node->fee\_paid, fee\_paid);

strcpy(new\_node->name, name);

new\_node->age = age;

strcpy(new\_node->organization, organization);

strcpy(new\_node->job, job);

new\_node->next = NULL;

Data\* ptr, \* phead, \* temp;

// Insert new node into last node

ptr = regi\_head->next;

while (ptr->next != NULL)

{

ptr = ptr->next;

}

ptr->next = new\_node;

// Bubble sort by tag

phead = regi\_head;

ptr = phead;

while (ptr->next != NULL)

{

if (ptr->tag > ptr->next->tag)

{

temp = ptr->next;

ptr->next = ptr->next->next;

temp->next = phead;

phead = ptr = temp;

continue;

}

ptr = ptr->next;

}

// Print added data and sorted linked list

ptr = regi\_head->next;

while (ptr != NULL)

{

printf("%d %s %s %s %d %s %s", ptr->tag, ptr->date, ptr->fee\_paid, ptr->name, ptr->age, ptr->organization, ptr->job);

// Data mark added

if (!strcmp(ptr->name, new\_node->name) && ptr->tag == new\_node->tag)

{

printf(" <===== added data");

}

printf("\n");

ptr = ptr->next;

}

printf("==========\n");

}

// ------------------------------------------

// P8-1 / 201834735 LeeJiheon

// Copy and compare

// ------------------------------------------

void copy\_is\_vaild(Data\* regi\_data, int data\_num)

{

printf("\n===P8-1===\n");

// Currently, the data in the array are sorted by age in 7-1.

FILE\* copyFile = fopen("8-1.txt", "w");

char checksum = '\0';

// Create checksum insert location on first line.

fprintf(copyFile, "%s\n", "00");

// 5 most recent data names in array

// Currently data\_num is a number containing the last null character(= '\0'), so -1.

for (int i = data\_num - 1; i > data\_num - 6; i--)

{

fprintf(copyFile, "%s\n", regi\_data[i].name);

int j = 0;

while (regi\_data[i].name[j] != '\0')

{

// using bitwise ExclusiveOR

checksum ^= regi\_data[i].name[j];

j++;

}

}

// Move the file pointer position to the start point. And Overwrite checksum value.

fseek(copyFile, 0, SEEK\_SET);

fprintf(copyFile, "%d\n", checksum);

fclose(copyFile);

compare\_checksum();

}

void compare\_checksum()

{

FILE\* readFile = fopen("8-1.txt", "r");

char str[25];

char read\_checksum = '\0', calculate\_checksum = '\0';

int line = 1;

while (fgets(str, 25, readFile) != NULL)

{

// Since there is a checksum on the first line, it counts by excluding the first line.

if (line != 1)

{

int i = 0;

// Currently, str contains '\n', so exclude it.

while (str[i] != '\n')

{

// using bitwise ExclusiveOR

calculate\_checksum ^= str[i];

i++;

}

}

else

{

int temp = atoi(str);

read\_checksum += temp;

}

line++;

}

fclose(readFile);

printf("Checksum attached to the copy : %d\n", read\_checksum);

printf("Checksum calculated from copied data : %d\n\n", calculate\_checksum);

if (read\_checksum == calculate\_checksum)

{

printf("The result of comparing %d(attached) and %d(calculated) is the same.\n", read\_checksum, calculate\_checksum);

}

else

{

printf("The result of comparing %d(attached) and %d(calculated), they are not equal.\n", read\_checksum, calculate\_checksum);

}

printf("==========\n");

}

Result screen capture

P1-1



P1-2



P2-1



P2-2



P3-1



P4-1



P6-1



P6-2

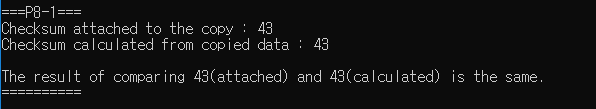


P7-1



P7-2

P8-1



Contribution percentage

**202035318 – Kim wonjong (25%)**

Setup, P3-1, P4-1 problem solving steps, source code

**202035352 – Ahn hyunjin (25%)**

P1-1, P1-2, P2-1, P2-2 problem solving steps, source code

**201834735 – Lee jiheon (25%)**

P7-1, P7-2, P8-1 problem solving steps, source code

**201835546 – Ham geonwook (25%)**

P5-1, P6-1, P6-2 problem solving steps, source code