**(1) Setup**

**Step 1: understand the problem**

We need to organize the information in the text file into an array and a ‘linked list’.

In ‘registration\_data’ text file, there are the information such as tag#, date registered, fee-paid, name, age, organization and job.

For example, write the information ‘6, 2020-06-04, yes, Bobby Anderson, 33, McGill University, engineer’ in the one line and must be separted by ‘/’.

Then, put the information into an array and a ‘linked list’.

\*\* Information Detail \*\*

**tag #** -> There should be no duplicates among the 30 data.

**date registered** -> write the date in ‘yyyy-mm-dd’ format.

**fee-paid** -> if fee-paid, then write the ‘yes’, otherwise, write the ‘no’.

**name** -> write the first and last name within 25 characters.

**age** -> write the age as an integer.

**organization** -> write the organization either ‘company’ or ‘university’ with the full name within 30 characters.

**job** -> write the job one of ‘student’, ‘professor’, ‘staff’, ‘executive’, ‘engineer’, or ‘marketer’ within 15 characters.

**Step 2: outline a solution**

Get the information by the file and loop through the information to save in the array and linked list for 30 times.

1) make the function, the information into the array using the function called ‘file\_line\_input’.

2) make the function, put the information into the linked list using the function called ‘add\_node’.

3) make the function, using ‘file\_line\_input’ and ‘add\_node’ to save real data. It is called ‘setup’ function.

**Step 3: form a program structure**

**‘file\_line\_input’**

for loop (0 through 30)

find & save the data in the file

**‘add\_node’**

using the node and pointer ‘next’ to find & save the data.

**‘setup’**

open the file ‘registration\_data.txt’ and read it.

for loop (0 through 30)

find & save the data in the array

for loop (0 through 30)

find & save the data in the linked list

**‘main’**

Using the function ‘setup’.

->Read data, store in array and linked list.

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

**‘file\_line\_input’**

for ( j=0; j<30; j++)

{

for (i=0; i<strlen(Fline\_Line[j] + 1; i++)

{

/\* find the ‘/’ and save it \*/

}

}

Put the correct value in the array.

**‘add\_node’**

Using the pointer ‘next’ and save the information

**‘setup’**

FILE \*fp1 = fopen (“registration\_data.txt”, “r” );

For (i=0; i<30; i++)

{

/\* get the data from a file to array \*/

}

//Use the function, file\_line\_input.

For (z=0; z<30; z++)

{

//Use the function, add\_node

}

**‘main’**

//use the function, setup

**(2) Search, Reorganize**

**Step 1: understand the problem**

In the first step, we save the data.

In this step, search the “Choi” and “Gachon University” in the array and the linked list

* If found the information, print all information about the person.

And then sort the data in the array by the tag# and age in the file ‘sorted data’.

**Step 2: outline a solution:** Find the “Choi”, and “Gachon University”.

1) make the function, search the data of name and print all information using the array about the persons. Function name is search\_name.

2) make the function, search the data of name and print all information using the linked list about the persons. Function name is search\_name2.

3) make the function, search the data of name of university and print all information using the array about the persons. Function name is search\_uni.

4) make the function, make the function, search the data of name of university and print all information using the linked list about the persons. Function name is search\_uni2.

5) make the function, it is to sort the information according to the ‘tag’ order using the array. Function name is sort\_array\_tag.

6) make the function, make the linked list that sorted by tag order. And print the linked list data.

Function name is create\_sort\_list.

7) make the function, it is to sort the information according to the ‘age’ order using the array. Function name is sort\_array\_age.

8) make the function, enter the data sorted in the order of tag and age in the file.

Function name is write\_sorted\_file.

**Step 3: form a program structure**

**search\_name**

for loop (0 through 30)

find the name and print all information about the persons

**search\_name2**

using the node/ find the name and print all information about the persons

**search\_uni**

for loop (0 through 30)

find the university and print all information about the persons

**search\_uni2**

using the node/ find the university and print all information about the persons

**sort\_array\_tag**

for loop (0 through 30)

store file\_line contents in file\_line1

for loop (0 through 30)

extracts person data from the line and store data

for loop (0 through 30)

sort by tag order

**create\_sort\_list**

for loop (0 through 30)

make linked list that sorted by tag order.

While (curr1 !=NULL)

Using the node, and print liked list data

**sort\_array\_age**

for loop (0 through 30)

sort by age order

for loop (0 through 30)

print the sorted data

**write\_sorted\_file**

read the file “sorted data.txt”

for loop (0 through 30)

enter data sorted in the order of tag in the file

for loop (0 through 30)

enter data sorted in the order of age in the file

**main**

using the function ‘search\_name’, ‘search\_name2’, ‘search\_uni’, ‘search\_uni2’, ‘sort\_array\_tag’, ‘sort\_array\_age’, ‘create\_sort\_list’, and ‘write\_sorted\_file’.

->Read the data and search the name and university.

->Sorting data and write in the file.

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

**search\_name**

for (j=0; j<30; j++)

{

for (z=0; z<strlen((P1+j)->name); z++)

{

/\* Find the name and print all information about the persons using the array. \*/

}

}

**search\_name2**

while (target != NULL)

{

For (z=0; z<strlen(target -> name); z++)

{

/\* Find the name and print all information about the persons using the node. \*/

}

}

**search\_uni**

for (j=0; j<30; j++)

{

for (z=0; z<strlen((P1+j)->university); z++)

{

/\* Find the university and print all information about the persons using the array. \*/

}

}

**search\_uni2**

while (target != NULL)

{

For (z=0; z<strlen(target -> university); z++)

{

/\* Find the university and print all information about the persons using the node. \*/

}

}

**sort\_array\_tag**

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

{

/\* store file\_line contents in file\_line1 \*/

}

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

{

/\*find the ‘/’ and extracts person data from the line and store data\*/

}

For (i=0;i<30;i++)

{

For (j=0; j<30-I; j++)

{

/\*sort by tag order \*/

}

/\*sort by tag order \*/

}

**create\_sort\_list**

for (z=0; z<30; z++)

{

/\*make linked list that sorted by tag order\*/

}

While (curr1 != NULL)

{

/\* print the all information \*/

}

**sort\_array\_age**

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

{

For (j=0; j<30-i; j++)

{

/\*sort by age order \*/

}

/\*sort by age order \*/

}

/\* print the all information \*/

**write\_sorted\_file**

FILE \*fp2 = fopen(“Sorted data.txt”, “w”);

For (i=0; i<30; i++)

{

/\*enter data sorted in the order of tag in the file\*/

}

For (i=0; i<30; i++)

{

/\*enter data sorted in the order of age in the file\*/

}

**main**

using the function ‘search\_name’, ‘search\_name2’, ‘search\_uni’, ‘search\_uni2’, ‘sort\_array\_tag’, ‘sort\_array\_age’, ‘create\_sort\_list’, and ‘write\_sorted\_file’.

**(2) Update**

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

Step 1 : understand the problem

-Delete data of Choi

-print data after delete choi’s data

Step 2: outline a solution

Loop through size of struct\_array\_num

if name is ‘Choi’ delete that data

print data after delete

Loop through ~curr!=NULL

if name is ‘Choi’ delete that data

print data after delete

Step 3: form a program structure

Main(function call)

remove\_struct(sort\_tag);

remove\_linkedlist(oldnode, curr1, &id\_num, head1);

function definition

Delete data from person with last name Choi

Print data after delete

Step 4: write a program outline (pseudo code)

while loop(curr!=NULL){

/\* if name is ‘Choi’ delete that data \*/

/\* print data after delete \*/

}

for loop ( 0 ~ struct\_array\_num) ){

/\* if name is ‘Choi’ delete that data \*/

/\* print data after delete \*/

}

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

Step 1 : understand the problem

-Save data to be added

-Store data in the order of tag numbers(in array, in linked list)

-print data

Step 2: outline a solution

store data to be added

Loop through size of struct\_array\_num

store data in array

print data

Loop through ~curr!=NULL

store data in array

print data

Step 3: form a program structure

Main(function call)

Add\_Array(sort\_tag, struct\_array\_num);

Add\_linked(sort\_tag, oldnode, head1, curr1);

function definition

Save data to be added

Store data in the order of tag numbers(in array, in linked list)

Step 4: write a program outline (pseudo code)

/\* Save data to be added \*/

while loop(curr!=NULL){

/\* Compare tag number and save in order \*/

/\* print data \*/

}

for loop ( 0 ~ struct\_array\_num) ){

/\* Compare tag number and save in order \*/

/\* print data \*/

}

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

Step 1 : understand the problem

-compute the checksum of the data’s name using bitwise XOR and attach it copy file

-compute the checksum int copied data, and compare it against the checksum in the original data

-confirm whether two data are the same

Step 2: outline a solution

Loop through size of struct\_array

Find checksum (Add all byte values of the name)

Checksum make checksum bite (checksum change two's complement)

Save checksum(changed two’s complement) in copy file

Loop through size of struct\_array

Obtain checksum value and name from copy file

Add all byte values of the name obtained.

Add checksum and name byte value

Change the above value to the two’s complement form

If above value is 0, data is same

Step 3: form a program structure

Main(function call)

Checksum(struct\_array\_num, sort\_tag);

Checksum Function

for loop(0~size of struct)

checksum change two complement form

save checksum and name in copy file

for loopt(0~size of struct)

Obtain checksum value and name from copy file

add two value and change two complement form

if above value is 0, data is same

Step 4: write a program outline (pseudo code)

for(i=0; i<struct\_array\_num; i++){

for(j=0; j<strlen(sort\_tag[i].name),j++){

/\* Find checksum (add all name’s bit value) \*/

}

/\* Change the checksum to two’s complement form and save in copy file\*/

}

for(i=0;i<struct\_array\_num;i++){

/\* Add checksum and bit value the name in copy file and add two value \*/

}

/\* if value is 0, data is same \*/

Source code

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct person {

int tag;

char date[20];

char fee\_paid[10];

char name[20];

int age;

char university[40];

char job[20];

};

struct NODE {

struct NODE\* next;

int tag;

char date[20];

char fee\_paid[10];

char name[20];

int age;

char university[40];

char job[20];

};

void File\_Line\_input(char File\_Line[][100], struct person\* input\_struct)//data\_store

{

char inf[7][40];

int count = 0;

int chn = 0;

int i, j;

for (j = 0; j < 30; j++)

{

for (i = 0; i < strlen(File\_Line[j]) + 1; i++)

{

if (File\_Line[j][i] == 47)

{

inf[count][chn] = 0;

count++;

chn = 0;

}

else if (File\_Line[j][i] != 47)

{

inf[count][chn] = File\_Line[j][i];

chn++;

}

else if (File\_Line[j][i] == 0)

{

inf[count][chn] = 0;

}

}

input\_struct[j].tag = atoi(inf[0]);

strcpy(input\_struct[j].date, inf[1]);

strcpy(input\_struct[j].fee\_paid, inf[2]);

strcpy(input\_struct[j].name, inf[3]);

input\_struct[j].age = atoi(inf[4]);

strcpy(input\_struct[j].university, inf[5]);

strcpy(input\_struct[j].job, inf[6]);

count = 0;

chn = 0;

}

}

void add\_node(struct NODE\* target, int tag, char date[], char fee\_paid[], char name[], int age, char uni[], char job[])

{

//Add node function

struct NODE\* newNode = (struct NODE\*)malloc(sizeof(struct NODE));

newNode->next = target->next;

newNode->tag = tag;

strcpy(newNode->date, date);

strcpy(newNode->fee\_paid, fee\_paid);

strcpy(newNode->name, name);

newNode->age = age;

strcpy(newNode->university, uni);

strcpy(newNode->job, job);

target->next = newNode;

}

void remove\_node(char name[], struct NODE\* target, struct NODE\* oldnode, int\* num)

{

//Remove node function

int i, j, z;

int check = 0;

int count = 0;

for (z = 0; z < strlen(target->name); z++)

{

if (target->name[z] == name[check])

{

if (check == strlen(name) - 1)

{

if (target->name[z + 1] == name[check + 1])

{

count++;

printf("\nDeleted Node : %s", target->name);

oldnode->next = target->next;

\*num = 100;

free(target);

break;

}

}

else

{

check++;

}

}

else

{

check = 0;

}

}

check = 0;

}

//원현아//

void setup(char File\_Line[][100], struct person\* p1, struct NODE\* address)

{

FILE\* fp1 = fopen("registraion\_data.txt", "r");

int i;

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

{

fgets(File\_Line[i], sizeof(File\_Line[i]), fp1);//save 30 lines of content in a file

}

fclose(fp1);

File\_Line\_input(File\_Line, p1); //process dataand store it in a struct array

int z;

for (z = 0; z < 30; z++) //store data in linked list

{

add\_node(address, p1[z].tag, p1[z].date, p1[z].fee\_paid, p1[z].name, p1[z].age, p1[z].university, p1[z].job);

address = address->next;

}

}

//원현아//

void search\_name(struct person\* P1)

{

char name[31];

printf("Search name :");

scanf("%[^\n]", name); //read name

printf("\nstruct name search");

printf("\n");

int i, j, z;

int check = 0;

int count = 0;

for (j = 0; j < 30; j++) //Search data of name that read

{

for (z = 0; z < strlen((P1 + j)->name); z++)

{

if ((P1 + j)->name[z] == name[check])

{

if (check == strlen(name) - 1)

{

if ((P1 + j)->name[z + 1] == name[check + 1]) //if found, print all information about the persons

{

count++;

printf("[%d] %d %s %s %s %d %s %s", count, (P1 + j)->tag, (P1 + j)->name, (P1 + j)->date, (P1 + j)->fee\_paid, (P1 + j)->age, (P1 + j)->university, (P1 + j)->job);

break;

}

}

else

{

check++;

}

}

else

{

check = 0;

}

}

check = 0;

}

}

//원현아//

void search\_name2(struct NODE\* target)

{

char name[31];

printf("\nSearch name :");

scanf(" %[^\n]", name); //read name

printf("\nlinked list name search");

printf("\n");

int i, j, z;

int check = 0;

int count = 0;

printf("\n");

while (target != NULL)

{

for (z = 0; z < strlen(target->name); z++)//Search data of name that read

{

if (target->name[z] == name[check])

{

if (check == strlen(name) - 1)

{

if (target->name[z + 1] == name[check + 1])//if found, print all information about the persons

{

count++;

printf("[%d] %d %s %s %s %d %s %s", count, target->tag, target->name, target->date, target->fee\_paid, target->age, target->university, target->job);

break;

}

}

else

{

check++;

}

}

else

{

check = 0;

}

}

check = 0;

target = target->next;

}

}

//원현아//

void search\_uni(struct person\* P1)

{

char uni[30];

printf("\nSearch university's name :");

scanf(" %[^\n]", uni); //read university

int i, j, z;

int check = 0;

int count = 0;

printf("\n");

for (j = 0; j < 30; j++)

{

for (z = 0; z < strlen((P1 + j)->university); z++)//Search data of university that read

{

if ((P1 + j)->university[z] == uni[check])

{

if (check == strlen(uni) - 1)//if found, print all information about the persons

{

count++;

printf("[%d] %d %s %s %s %d %s %s", count, (P1 + j)->tag, (P1 + j)->name, (P1 + j)->date, (P1 + j)->fee\_paid, (P1 + j)->age, (P1 + j)->university, (P1 + j)->job);

break;

}

else

{

check++;

}

}

else

{

check = 0;

}

}

check = 0;

}

}

//원현아//

void search\_uni2(struct NODE\* target)

{

char uni[30];

printf("\n\nSearch university's name :");

scanf(" %[^\n]", uni); //read university

int i, j, z;

int check = 0;

int count = 0;

printf("\n");

while (target != NULL)

{

for (z = 0; z < strlen(target->university); z++)//Search data of university that read

{

if (target->university[z] == uni[check])

{

if (check == strlen(uni) - 1)//if found, print all information about the persons

{

count++;

printf("[%d] %d %s %s %s %d %s %s", count, target->tag, target->name, target->date, target->fee\_paid, target->age, target->university, target->job);

break;

}

else

{

check++;

}

}

else

{

check = 0;

}

}

check = 0;

target = target->next;

}

}

//원현아//

void Sort\_array\_tag(char File\_Line[][100], struct person\* sort\_tag, char File\_Line1[][100])

{

char num1[30][3];

int num[30];

int i, j;

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

{

strcpy(File\_Line1[i], File\_Line[i]); //Store File\_line contents in File\_Line1

}

for (i = 0; i < 30; i++)//Extracts person data from the line and store data

{

for (j = 0; j < 3; j++)

{

if (File\_Line[i][j] != 47)

{

num1[i][j] = File\_Line[i][j];

}

else if (File\_Line[i][j] == 47)

{

num1[i][j] = 0;

break;

}

}

num[i] = atoi(num1[i]);

}

int max;

int count;

int temp;

char temp1[100];

for (i = 0; i < 30; i++) //Sort by Tag Order

{

max = num[0];

for (j = 0; j < 30 - i; j++)

{

if (num[j] >= max)

{

max = num[j];

count = j;

}

else

{

continue;

}

}

temp = num[j - 1];

num[j - 1] = num[count];

num[count] = temp;

strcpy(temp1, File\_Line1[j - 1]);

strcpy(File\_Line1[j - 1], File\_Line1[count]);

strcpy(File\_Line1[count], temp1);

}

printf("\n");

File\_Line\_input(File\_Line1, sort\_tag);

printf("\n");

for (int z = 0; z < 30; z++)

{

printf("[Tag:%d] %d %s %s %s %d %s %s", z + 1, sort\_tag[z].tag, sort\_tag[z].date, sort\_tag[z].fee\_paid, sort\_tag[z].name, sort\_tag[z].age, sort\_tag[z].university, sort\_tag[z].job);

}

}

//김범진//

void Create\_sort\_list(struct person\* sort\_tag, struct NODE\* address1, struct NODE\* head1, struct NODE\* curr1)

{

int number = 0, z;

for (z = 0; z < 30; z++)//Make linked list that sorted by tag oreder

{

add\_node(address1, sort\_tag[z].tag, sort\_tag[z].date, sort\_tag[z].fee\_paid, sort\_tag[z].name, sort\_tag[z].age, sort\_tag[z].university, sort\_tag[z].job);

address1 = address1->next;

}

printf("\nPrint tag order linked list data\n");

number = 0;

curr1 = head1->next;

while (curr1 != NULL) //Print linked list data

{

number++;

printf("[Tag:%d] %d %s %s %s %d %s %s", number, curr1->tag, curr1->date, curr1->fee\_paid, curr1->name, curr1->age, curr1->university, curr1->job);

curr1 = curr1->next;

}

}

//김범진//

void Sort\_array\_age(char File\_Line[][100], struct person\* sort\_age, struct person\* p1)

{

int i, j;

int max;

int count;

int temp;

char temp1[100];

for (i = 0; i < 30; i++)//Sort by Age Order

{

max = p1[0].age;

for (j = 0; j < 30 - i; j++)

{

if (p1[j].age >= max)

{

max = p1[j].age;

count = j;

}

else

{

continue;

}

}

temp = p1[j - 1].age;

p1[j - 1].age = p1[count].age;

p1[count].age = temp;

strcpy(temp1, File\_Line[j - 1]);

strcpy(File\_Line[j - 1], File\_Line[count]);

strcpy(File\_Line[count], temp1);

}

File\_Line\_input(File\_Line, sort\_age);

printf("\n");

printf("\nPrint Age order struct array\n");

for (j = 0; j < 30; j++)

{

printf("[Age:%d] %d-%s-%s-%s-%d-%s-%s", sort\_age[j].age, sort\_age[j].tag, sort\_age[j].date, sort\_age[j].fee\_paid, sort\_age[j].name, sort\_age[j].age, sort\_age[j].university, sort\_age[j].job);

}

}

//김범진//

void Write\_Sorted\_File(struct person\* sort\_tag, struct person\* sort\_age, char File\_Line[][100], char File\_Line1[][100])

{

FILE\* fp2 = fopen("Sorted data.txt", "w");

int i, j;

fprintf(fp2, "------------------------------------Order-Tag------------------------------------\n\n");

for (i = 0; i < 30; i++)//Enter data sorted in the order of tag in the file

{

fprintf(fp2, "[%d] ", sort\_tag[i].tag);

fputs(File\_Line1[i], fp2);

}

fprintf(fp2, "\n------------------------------------Order-Age------------------------------------\n\n");

for (i = 0; i < 30; i++)//Enter data sorted in the order of age in the file

{

fprintf(fp2, "[%d] ", sort\_age[i].age);

fputs(File\_Line[i], fp2);

}

fclose(fp2);

}

//김범진//

int Array\_number = 0; //size of struct array//

void remove\_struct(struct person\* sort\_tag)

{

char remove\_name[30];

printf("\nSearch name that will be removed :");

scanf(" %[^\n]", remove\_name); //Read name that will be removed

int i, j, z;

int check = 0;

int count = 0;

int del\_num;

int num;

int struct\_num = 0;

for (j = 0; j < 30; j++)

{

for (z = 0; z < strlen((sort\_tag + j)->name); z++)

{

if ((sort\_tag + j)->name[z] == remove\_name[check])

{

if (check == strlen(remove\_name) - 1)

{

if ((sort\_tag + j)->name[z + 1] == remove\_name[check + 1])//If name found, Delete that person's data

{

del\_num = j;

printf("\nDeleted Name : %s", (sort\_tag + j)->name);

for (del\_num = j; del\_num < (sizeof(sort\_tag[0]) / sizeof(sort\_tag)); del\_num++)

{

memcpy(&sort\_tag[del\_num], &sort\_tag[del\_num + 1], sizeof(person));

}

count++;

j--;

break;

}

}

else

{

check++;

}

}

else

{

check = 0;

}

}

check = 0;

}

Array\_number = count;

printf("\nStruct Array (Choi's data removed)\n");

for (j = 0; j < sizeof(sort\_tag[0]) / sizeof(sort\_tag) - (Array\_number); j++)//Print struct array after removed

{

printf("[%d] %d-%s-%s-%s-%d-%s-%s", j + 1, sort\_tag[j].tag, sort\_tag[j].date, sort\_tag[j].fee\_paid, sort\_tag[j].name, sort\_tag[j].age, sort\_tag[j].university, sort\_tag[j].job);

}

Array\_number = j - 1;

}

//김범진//

void remove\_linkedlist(struct NODE\* oldnode, struct NODE\* curr1, int\* id\_num, struct NODE\* head1)

{

int i;

char remove\_name[30];

printf("\nSearch name that will be removed :");

scanf(" %[^\n]", remove\_name); //Read name that will be removed

printf("\nlinked list (Choi's data removed)");

while (curr1 != NULL)

{

remove\_node(remove\_name, curr1, oldnode, id\_num);//If name found, Delete that person's data

if (\*id\_num == 100)

{

curr1 = oldnode->next;

}

else

{

curr1 = curr1->next;

oldnode = oldnode->next;

}

\*id\_num = 0;

}

i = 0;

printf("\n\nlinked list after removed");

printf("\n");

curr1 = head1->next;

while (curr1 != NULL)//Print linked list after removed

{

i++;

printf("[%d] %d %s %s %s %d %s %s", i, curr1->tag, curr1->name, curr1->date, curr1->fee\_paid, curr1->age, curr1->university, curr1->job);

curr1 = curr1->next;

}

}

//김범진//

void Add\_Array(struct person\* sort\_tag, int struct\_array\_num)

{

int i, j;

//Store new people's data

sort\_tag[struct\_array\_num].tag = 2;

strcpy(sort\_tag[struct\_array\_num].date, "2001-11-01");

strcpy(sort\_tag[struct\_array\_num].fee\_paid, "yes");

strcpy(sort\_tag[struct\_array\_num].name, "beom Paik");

sort\_tag[struct\_array\_num].age = 21;

strcpy(sort\_tag[struct\_array\_num].university, "Gachon University");

strcpy(sort\_tag[struct\_array\_num].job, "student\n");

for (i = 0; i <= struct\_array\_num; i++)//Store new data in tag order

{

if (sort\_tag[struct\_array\_num].tag < sort\_tag[i].tag)

{

sort\_tag[struct\_array\_num + 1].tag = sort\_tag[struct\_array\_num].tag;

for (j = struct\_array\_num + 1; j >= i; j--)

{

memcpy(&sort\_tag[j], &sort\_tag[j - 1], sizeof(person));

}

memcpy(&sort\_tag[i], &sort\_tag[struct\_array\_num + 1], sizeof(person));

struct\_array\_num++;

break;

}

}

printf("\n\nStruct Array after Add \n");//Print struct array after add new data

for (j = 0; j <= struct\_array\_num; j++)

{

printf("[%d] %d-%s-%s-%s-%d-%s-%s", j + 1, sort\_tag[j].tag, sort\_tag[j].date, sort\_tag[j].fee\_paid, sort\_tag[j].name, sort\_tag[j].age, sort\_tag[j].university, sort\_tag[j].job);

}

}

//김범진//

void Add\_linked(struct person\* sort\_tag, struct NODE\* oldnode, struct NODE\* head1, struct NODE\* curr1)

{

int i;

//Store new people's data

sort\_tag[30].tag = 2;

strcpy(sort\_tag[30].date, "2001-11-01");

strcpy(sort\_tag[30].fee\_paid, "yes");

strcpy(sort\_tag[30].name, "beom Paik");

sort\_tag[30].age = 21;

strcpy(sort\_tag[30].university, "Gachon University");

strcpy(sort\_tag[30].job, "student");

oldnode = head1;

curr1 = head1->next;

i = 0;

while (curr1 != NULL)

{

i++;

if (sort\_tag[30].tag < curr1->tag) // Store new data in tag order

{

add\_node(oldnode, sort\_tag[30].tag, sort\_tag[30].date, sort\_tag[30].fee\_paid, sort\_tag[30].name, sort\_tag[30].age, sort\_tag[30].university, sort\_tag[30].job);

break;

}

else

{

curr1 = curr1->next;

oldnode = oldnode->next;

}

}

i = 0;

printf("\n\nlinked list after add");

printf("\n");

curr1 = head1->next;

int iu = 0;

while (curr1 != NULL)//Print linked list after add new data

{

iu++;

if (iu == 2)

printf("\n");

i++;

printf("[%d] %d %s %s %s %d %s %s", i, curr1->tag, curr1->name, curr1->date, curr1->fee\_paid, curr1->age, curr1->university, curr1->job);

curr1 = curr1->next;

}

}

//김범진//

void Checksum(int struct\_array\_num, struct person\* sort\_tag)

{

int total = 0;

int sum = { 0 };

FILE\* check = fopen("checksum.txt", "w");

int i, j, z;

for (i = 0; i < struct\_array\_num; i++)//Compute the checksum of the original data

{

sum = 0;

for (j = 0; j < strlen(sort\_tag[i].name); j++)

{

sum += sort\_tag[i].name[j];

}

total = sum;

total = total & 0xFF;

total = total ^ 255; //using bitwise ExclusiveOR

total = total + 1;

fprintf(check, "%d\n", total); //attach it to the copy.

fprintf(check, "%s\n", sort\_tag[i].name);

}

fclose(check);

FILE\* checkread = fopen("checksum.txt", "r");

int check\_count = 0;

int checknumber = 0;

char name2[20];

int sum1[30] = { 0 };

for (i = 0; i < struct\_array\_num; i++)

{

fscanf(checkread, "%d", &checknumber);

fscanf(checkread, " %[^\n]", name2);

for (j = 0; j < strlen(name2); j++)//Compute the checksum in the copied data

{

sum1[i] += name2[j];

}

checknumber = checknumber + sum1[i];//add checknumber in copy file and checksum

checknumber = checknumber & 0xff;

if (checknumber == 0)//if checknumber is 0, data is same

{

check\_count++; //if data is same, count plus 1

}

}

if (check\_count == struct\_array\_num)//if count equal size of struct, print "Two data is same" message

{

printf("\n\n[Copy Test] Two data is same");

}

else //else , print "Two data is not same" message

{

printf("\n\n[Copy Test] Two data is not same");

}

fclose(checkread);

}

int main()

{

struct person p1[30];

char File\_Line[30][100];

char File\_Line1[30][100];

int i, j, z;

struct NODE\* head = (struct NODE\*)malloc(sizeof(struct NODE));

head->next = NULL;

struct NODE\* address = (struct NODE\*)malloc(sizeof(struct NODE));

address = head;

struct NODE\* curr;

//Setup(Read data, store in array and linked list)//

setup(File\_Line, p1, address);

//Search for “Choi"(struct array, linked list)//

search\_name(p1);

curr = head->next;

search\_name2(curr);

//Search for all from Gachon University (struct array, linked list)//

search\_uni(p1);

curr = head->next;

search\_uni2(curr);

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

struct person sort\_tag[31];

Sort\_array\_tag(File\_Line, sort\_tag, File\_Line1);

//Create a linked list using the sorted data.//

struct NODE\* head1 = (struct NODE\*)malloc(sizeof(struct NODE));

head1->next = NULL;

struct NODE\* address1 = (struct NODE\*)malloc(sizeof(struct NODE));

address1 = head1;

struct NODE\* curr1 = head1->next;

struct NODE\* oldnode = (struct NODE\*)malloc(sizeof(struct NODE));

int id\_num = 0;

Create\_sort\_list(sort\_tag, address1, head1, curr1);

//Sort the data in the array in age group order//

struct person sort\_age[30];

Sort\_array\_age(File\_Line, sort\_age, p1);

//Write the sorted data to a text file.//

Write\_Sorted\_File(sort\_tag, sort\_age, File\_Line, File\_Line1);

//All“Choi”s canceled registration. Remove the data from(array, linked list)//

remove\_struct(sort\_tag);

oldnode = head1;

curr1 = head1->next;

remove\_linkedlist(oldnode, curr1, &id\_num, head1);

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

int struct\_array\_num = Array\_number;

Add\_Array(sort\_tag, struct\_array\_num);

Add\_linked(sort\_tag, oldnode, head1, curr1);

//Copy the most recent data in the array for transmission to a remote computer//

Checksum(struct\_array\_num, sort\_tag);

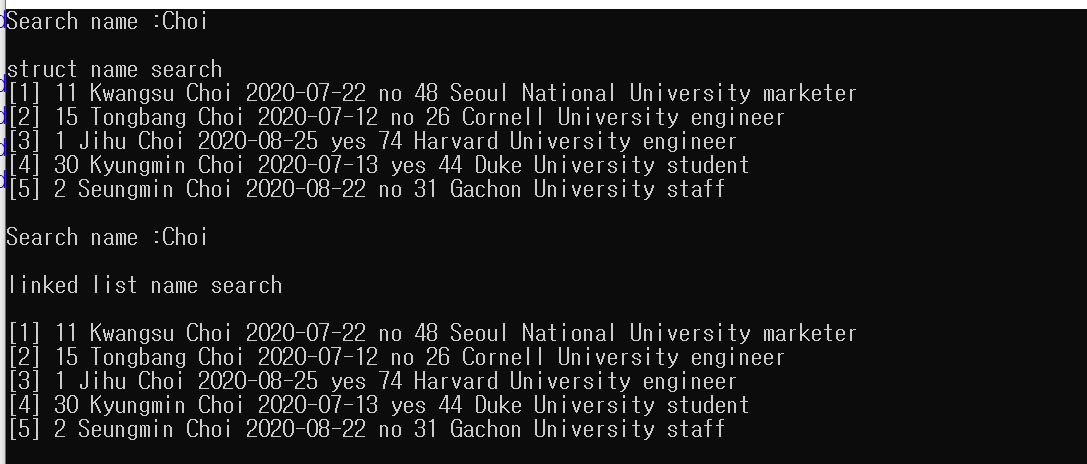
}

contribution

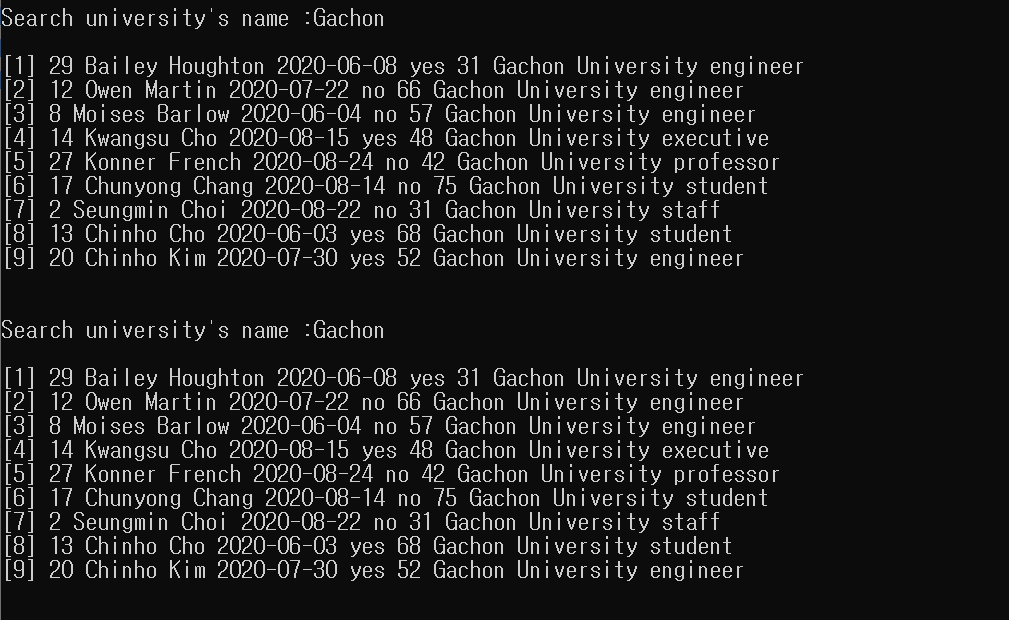
김범진 : 58%

원현아 : 42%

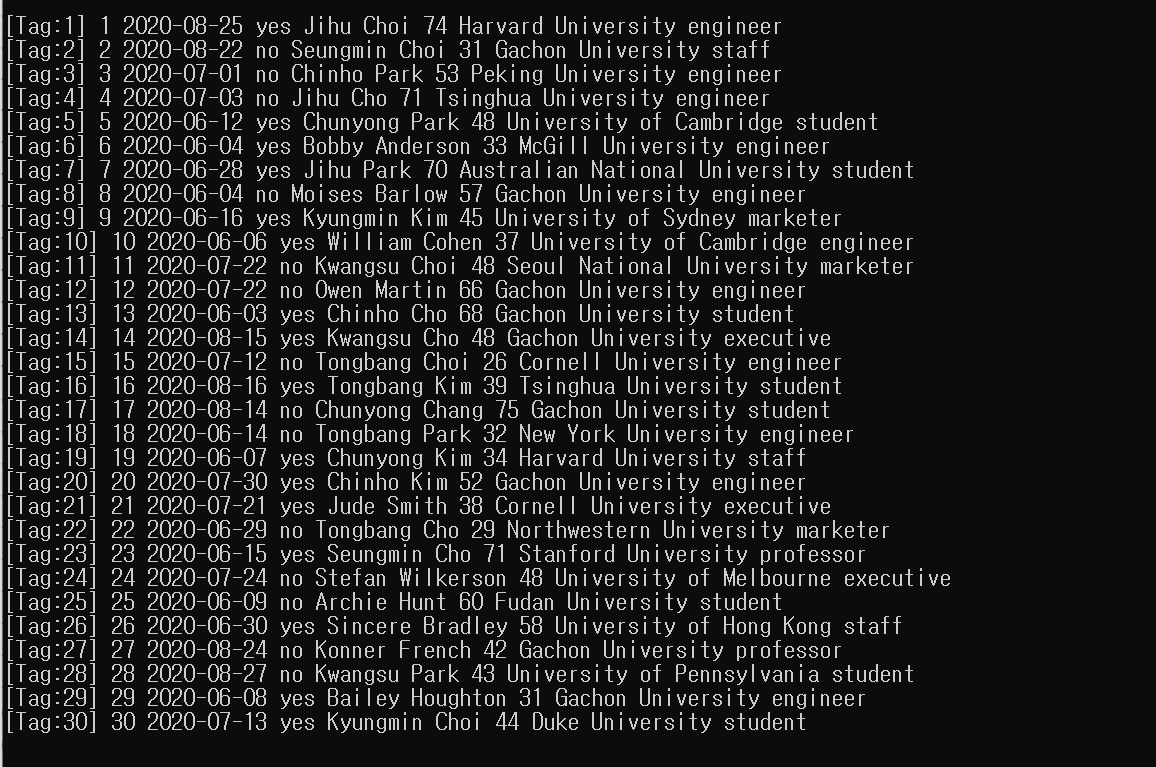
Search for “Choi”(struct array, linked list)



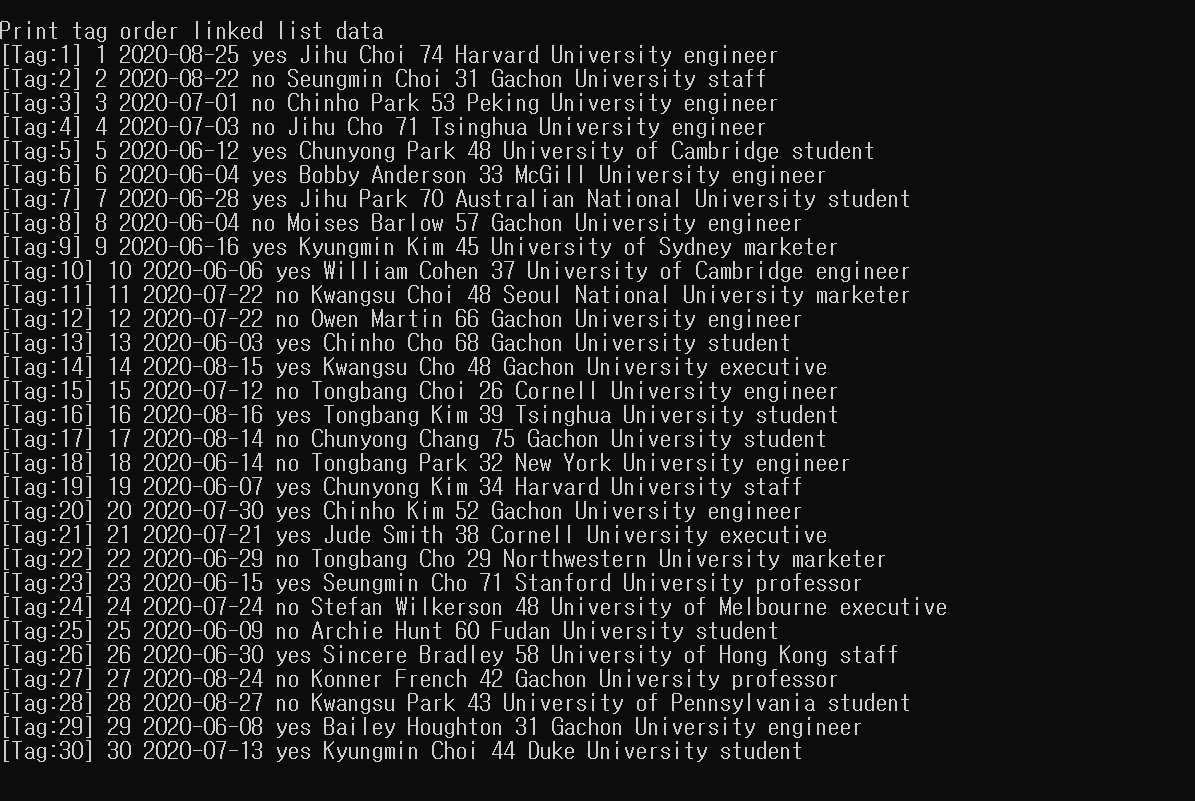
Search for all from Gachon University(struct array, linked list)



Sort the data in array in tag order

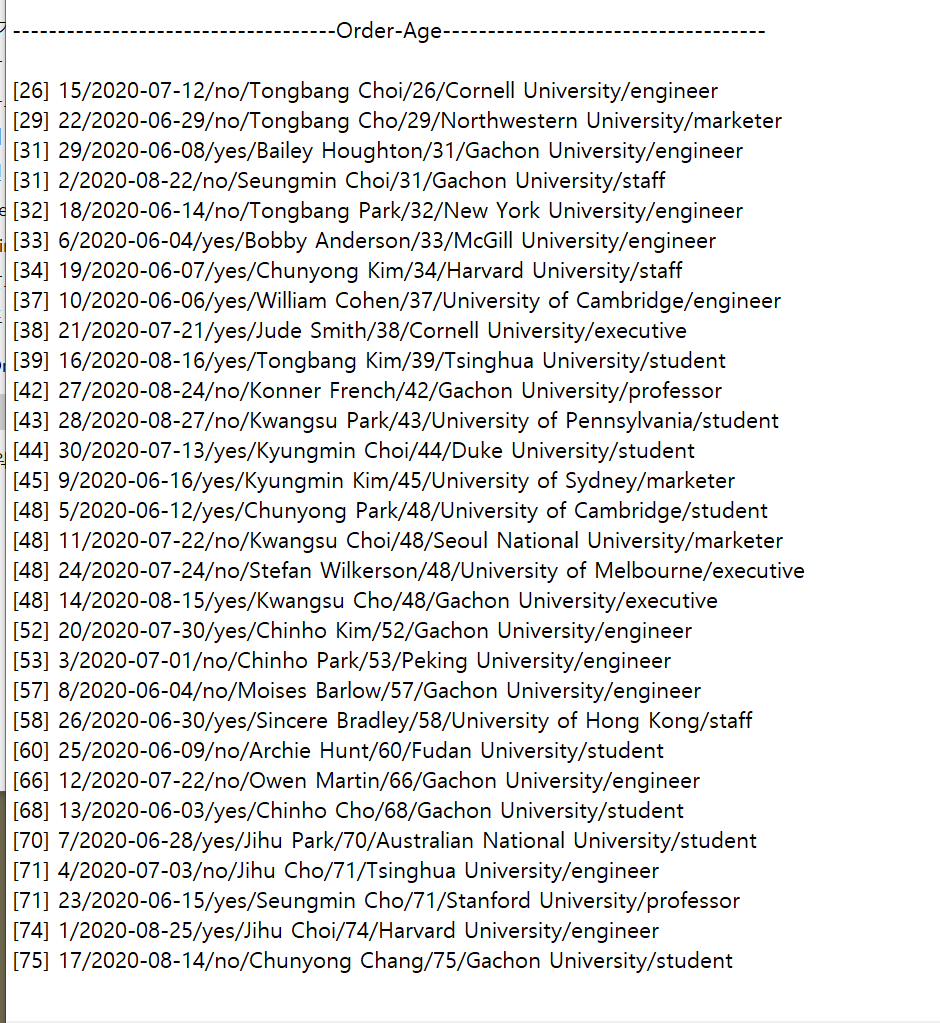


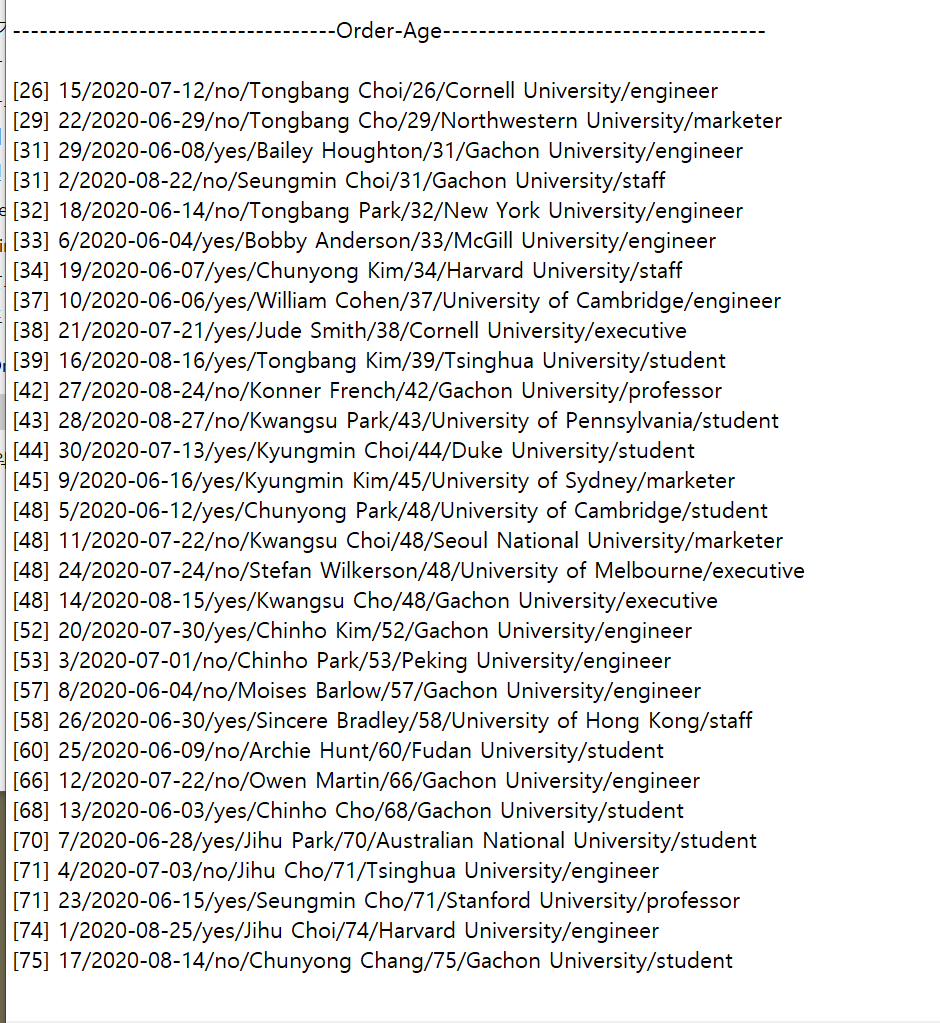
Create a linked list using sorted data



Sort the data in the array in age order



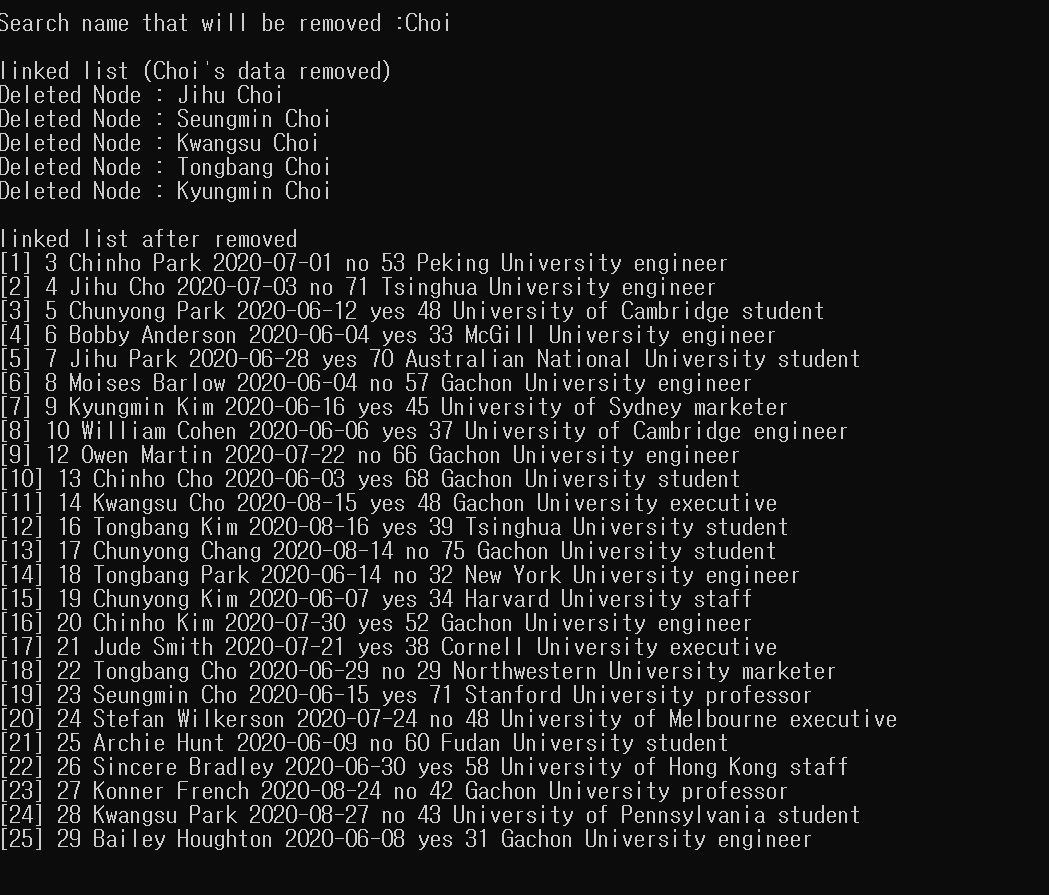
Write the sorted data to text file.



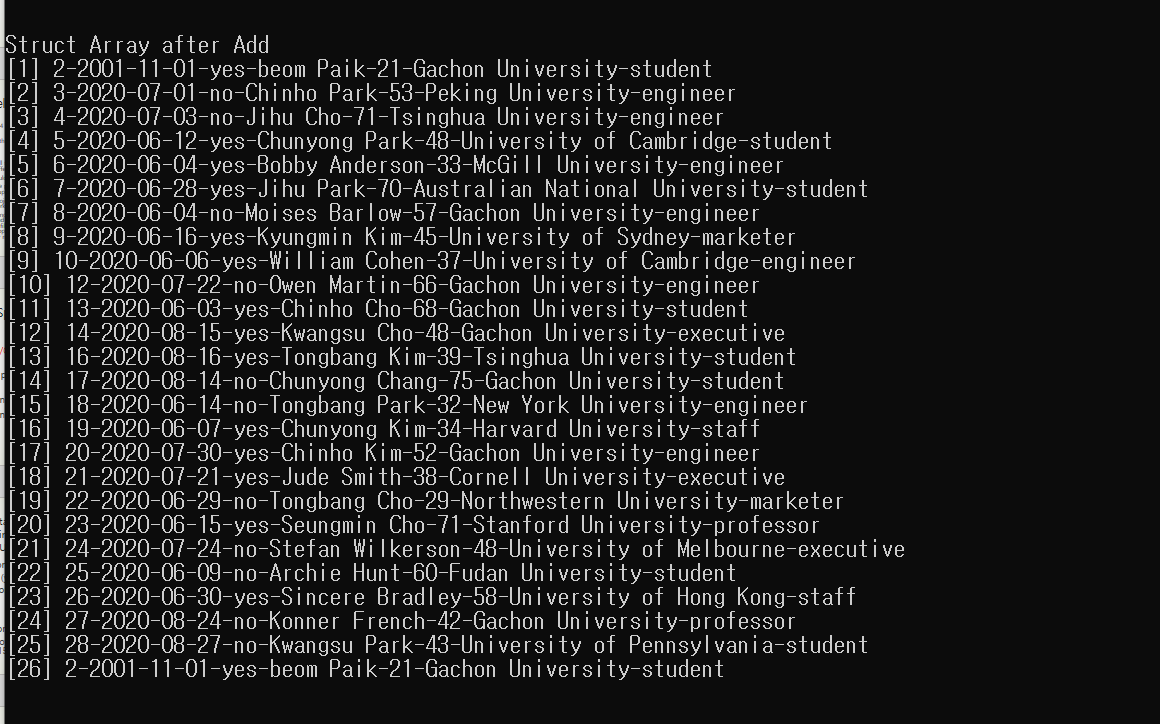
All “Choi”s cancled registration(struct)



All “Choi”s cancled registration(linked list)



One Paik registered late. Add the data(struct)



One Paik registered late. Add the data(linked list)



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

