ASSIGNMENT 2

B21946529

Fatih Pehlivan

03-12-2021

1-) Introduction: The Project wants me to make an employee registration system. When I add something or print something to the console I have some restrictions. For example sometimes I have to print employees by employee number ascending order,etc.

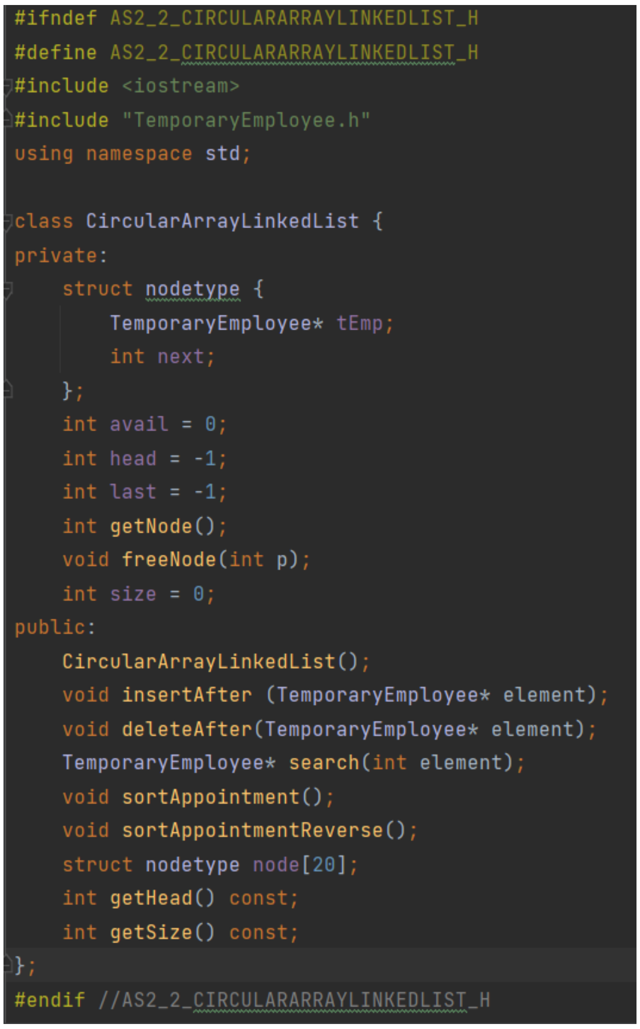
2-) Method: I use singly-circular-linkedlist and doubly linked list when I am listing employees (Temprory and Permanent) because we don't know where to add elements. In array we have to shift elements when add something to it. No shifting requires linked-list. Just add element in linked-list.

3-) Development:

3-1) Plan: Firstly, I implement linked-lists and try them it works true or false with integer values. Then I created my employee clasess. At least I make user interface.

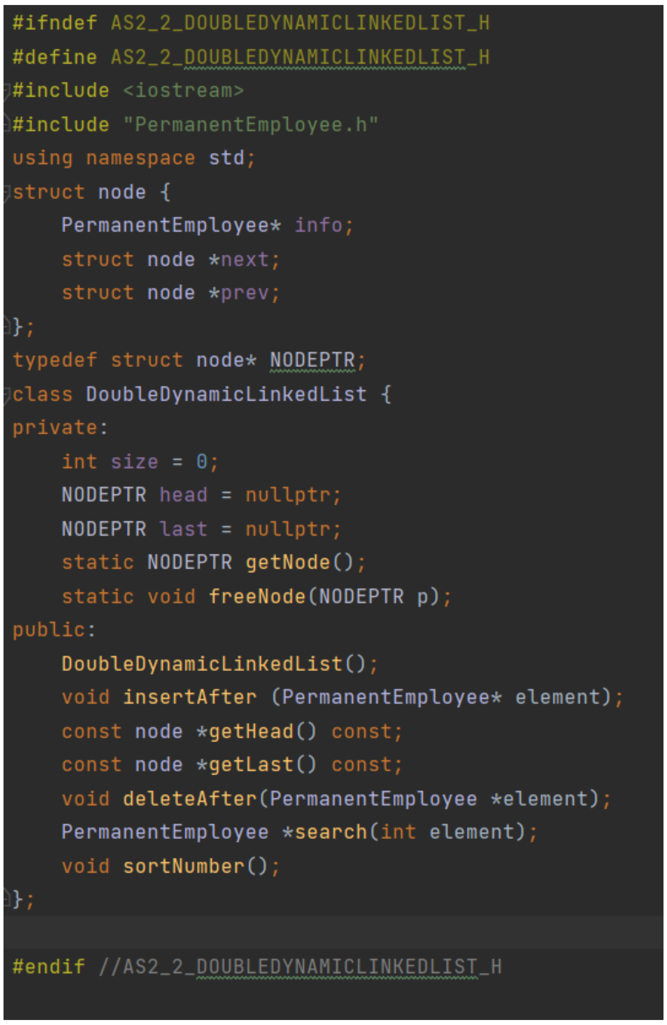
3-2) Analysis: Using linked-list is better than other data types such as array vector ect. The problem is making a an employee registration system.

3-3) Design: I choose linked-list because adding element is more efficent than arrays or vectors. Maybe lists can be used for this assignment. I eleminated lists because it has no iterator and linked-list is more efficent.



```cpp
#ifndef AS2_2_CIRCULARARRAYLINKEDLIST_H
#define AS2_2_CIRCULARARRAYLINKEDLIST_H
#include <iostream>
#include "TemporaryEmployee.h"
using namespace std;

class CircularArrayLinkedList {
private:
    struct nodetype {
        TemporaryEmployee* tEmp;
        int next;
    };
    int avail = 0;
    int head = -1;
    int last = -1;
    int getNode();
    void freeNode(int p);
    int size = 0;
public:
    CircularArrayLinkedList();
    void insertAfter (TemporaryEmployee* element);
    void deleteAfter(TemporaryEmployee* element);
    TemporaryEmployee* search(int element);
    void sortAppointment();
    void sortAppointmentReverse();
    struct nodetype node[20];
    int getHead() const;
    int getSize() const;
};
#endif //AS2_2_CIRCULARARRAYLINKEDLIST_H
```



```cpp
#ifndef AS2_2_DOUBLEDYNAMICLINKEDLIST_H
#define AS2_2_DOUBLEDYNAMICLINKEDLIST_H
#include <iostream>
#include "PermanentEmployee.h"
using namespace std;
struct node {
    PermanentEmployee* info;
    struct node *next;
    struct node *prev;
};
typedef struct node* NODEPTR;
class DoubleDynamicLinkedList {
private:
    int size = 0;
    NODEPTR head = nullptr;
    NODEPTR last = nullptr;
    static NODEPTR getNode();
    static void freeNode(NODEPTR p);
public:
    DoubleDynamicLinkedList();
    void insertAfter (PermanentEmployee* element);
    const node *getHead() const;
    const node *getLast() const;
    void deleteAfter(PermanentEmployee *element);
    PermanentEmployee *search(int element);
    void sortNumber();
};

#endif //AS2_2_DOUBLEDYNAMICLINKEDLIST_H
```

For circular-linked-list: I have a nodetype struct. It has temporary employee values and integer next array index.

Private variables and Functions:

I have some integer variables to define head, avail node index, last and size. I have 2 functions getNode and freeNode.

Head shows linked-list first element's index.

Avail shows avail node index

Last shows linked-list last element's index

Size shows the size of the array.

I change the variables after inserting or deleting or sorting the linked-list.

I use getNode to take a free area in the array.

I use freeNode function to make free which is deleted.

Public Functions:

I used CicrcularArrayLinkedList for constructor. I define my node array.

insertAfter add element in the linked-list by ascending order by employee number.

deleteAfter remove the element which you choose.

search function searchs element according to temproryEmployeeNumber.

sortAppointment sorts the list according to appointment date old to new.

sortAppointmentReverse sorts the list according to appointments new to old.

node is my struct.

getHead returns to head index.

getSize returns the size.

For doubly-linked-list: I have a node struct. It has permamant employee values and node pointer next. I define node pointer as NODEPTR

Private variables and Funcions:

I have a integer variable to define size.

I have some NODEPTR variables to define head, last. I have 2 functions getNode and freeNode.

Size shows size of linked-list.

head shows first NODEPTR of linked-list.

last shows last NODEPTR of linked-list.

I change the variables after inserting or deleting or sorting the linked-list.

I use getNode to delete pointer.

I use freeNode function to make free which is deleted.

Public Functions:

I used DoubleDynamicLinkedList for constructor.

insertAfter add element in the linked-list by old to new by appointment date.

deleteAfter remove the element which you choose.

search function searches an element according to permanentEmployeeNumber.

sortNumber sorts the list according to employee number ascending order.

getHead returns to head NODEPTR.

getLast returns the last NODEPTR.

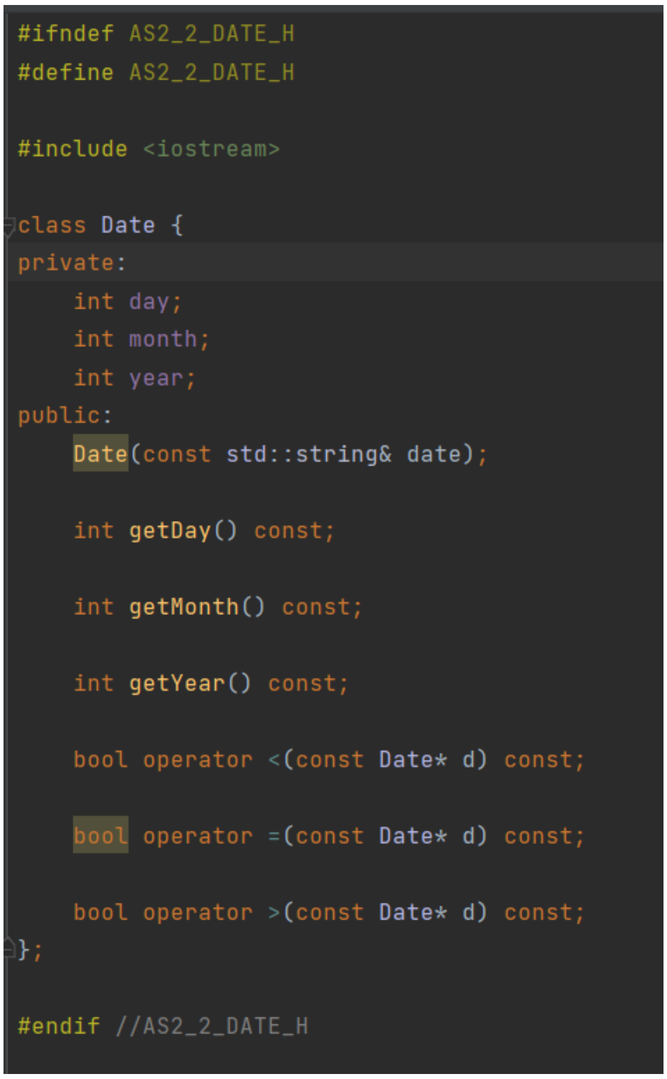
3-4) Implementation:

Firstly I created 2 circular linked lists. The first one is original linked-list, the second one is after sorting elements I used it. Then created 2 doubly linked lists, the same logic with the circular. The others are in a loop. I print the main menu and get the number of orders from the user. I take some information from the user such as name, surname, birthdate... (detailed explanation is in the next part)

After making the command, print the main board again and so on.

3-5) Programmer Catalog:

This assignment takes approximately 3 weeks. Other developers reuse my code, my functions run correctly. There is no error check in this assignment. Watch out the inputs are valid.



```
#ifndef AS2_2_DATE_H
#define AS2_2_DATE_H

#include <iostream>

class Date {
private:
    int day;
    int month;
    int year;
public:
    Date(const std::string& date);

    int getDay() const;

    int getMonth() const;

    int getYear() const;

    bool operator <(const Date* d) const;

    bool operator =(const Date* d) const;

    bool operator >(const Date* d) const;
};

#endif //AS2_2_DATE_H
```

This is my Date class. When you implement this class you have to give it a date. (the format should be dd-mm-yyyy).

Day shows the day of the date. Month shows the month of the date. Year shows the year of the date.

getDay returns the day of the date.

getMonth returns the month of the date.

getYear returns the year of the date.

opeartor < compare its date with other date and returns true if its date is older.

opeartor > compare its date with other date and returns true if its date is newer.

opeartor = compare its date with other date and returns true if dates are equal.

This is my employee class. Here are my private values.

employeeNumber shows employee number.

employeeType shows employee type.

name shows employee name.

surname shows employee surname.

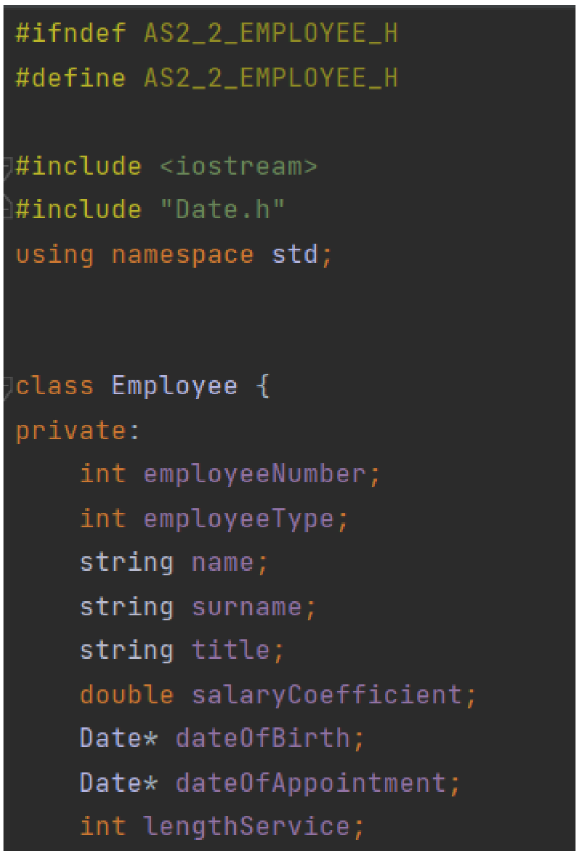
salaryCoefficent shows employee salary coefficient.

dateOfBirth shows employee birth date.

dateOfAppointment shows employee date of appointment.

lengthService shows employee lengthService.

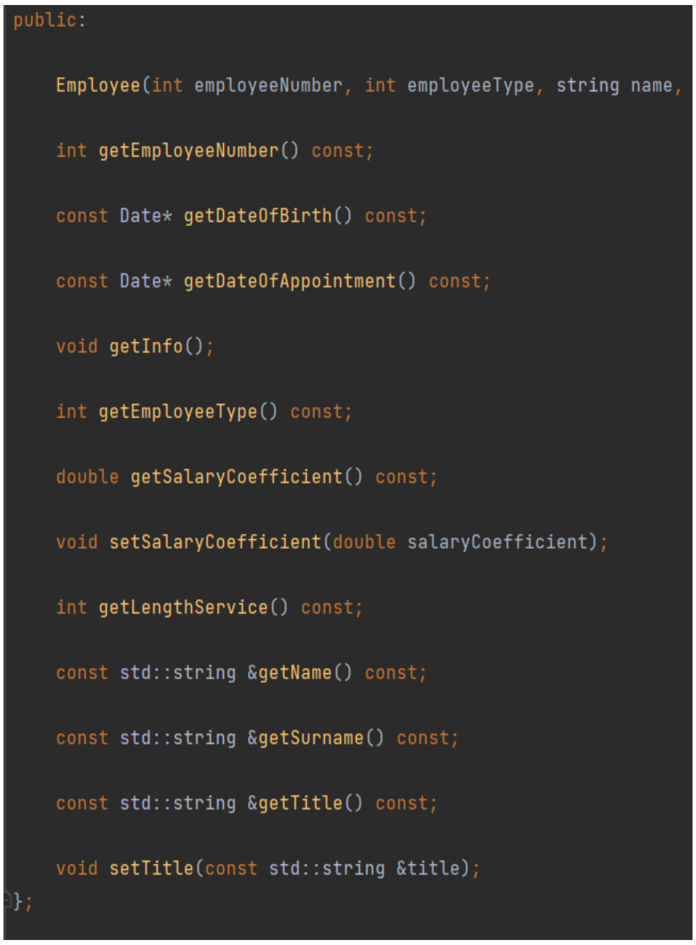
Default of lengthService is zero



```
#ifndef AS2_2_EMPLOYEE_H
#define AS2_2_EMPLOYEE_H

#include <iostream>
#include "Date.h"
using namespace std;


class Employee {
private:
    int employeeNumber;
    int employeeType;
    string name;
    string surname;
    string title;
    double salaryCoefficient;
    Date* dateOfBirth;
    Date* dateOfAppointment;
    int lengthService;
```



```cpp
public:

    Employee(int employeeNumber, int employeeType, string name,

    int getEmployeeNumber() const;

    const Date* getDateOfBirth() const;

    const Date* getDateOfAppointment() const;

    void getInfo();

    int getEmployeeType() const;

    double getSalaryCoefficient() const;

    void setSalaryCoefficient(double salaryCoefficient);

    int getLengthService() const;

    const std::string &getName() const;

    const std::string &getSurname() const;

    const std::string &getTitle() const;

    void setTitle(const std::string &title);
};
```

Here are my public methods.

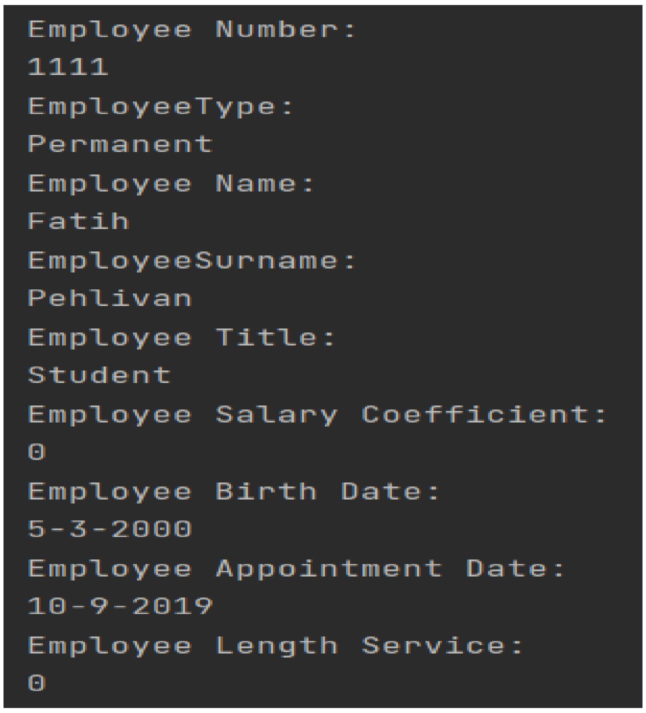
I cannot screen shoot all Employee arguments but it is the same as private values.

Getter functions (except getInfo) returns the private values according to function name.

Setter functions set private values according to the new values.

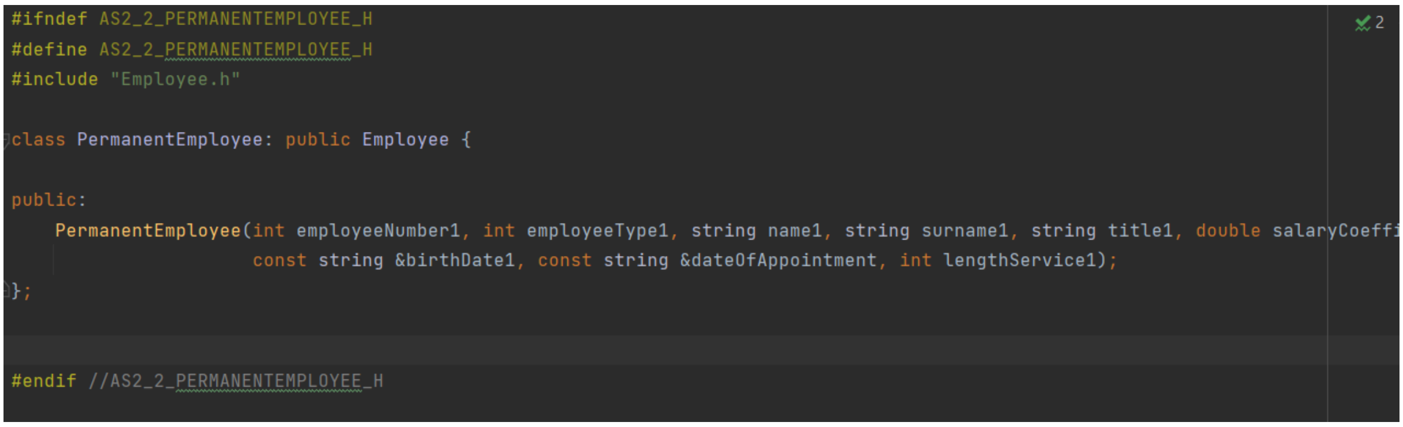
getInfo prints employee info in the console.

Like:



```
Employee Number:
1111
EmployeeType:
Permanent
Employee Name:
Fatih
EmployeeSurname:
Pehlivan
Employee Title:
Student
Employee Salary Coefficient:
0
Employee Birth Date:
5-3-2000
Employee Appointment Date:
10-9-2019
Employee Length Service:
0
```

My PermanentEmployee and TemprorayEmployee classes are only subclass of Employee class.

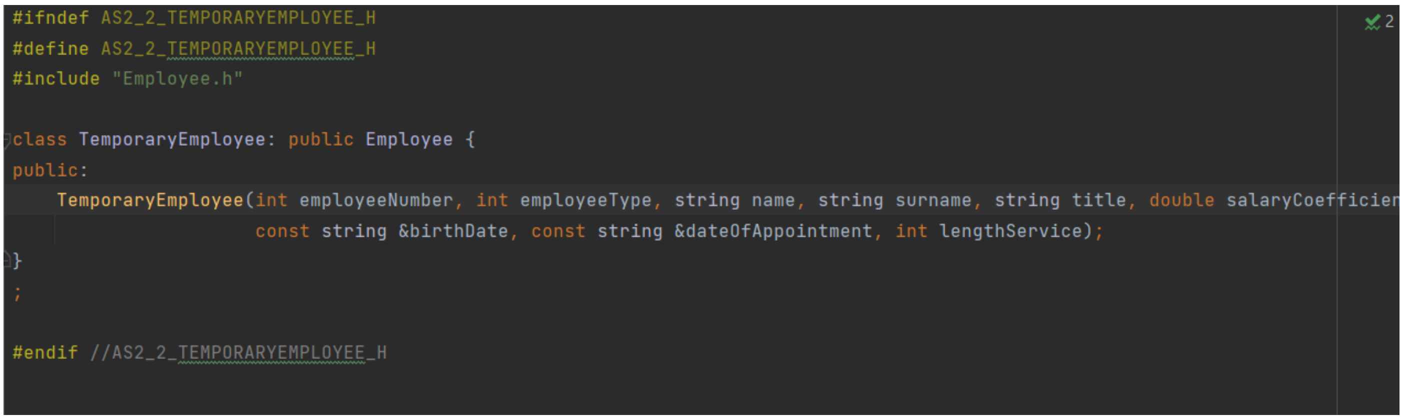


```cpp
#ifndef AS2_2_PERMANENTEMPLOYEE_H
#define AS2_2_PERMANENTEMPLOYEE_H
#include "Employee.h"

class PermanentEmployee: public Employee {

public:
    PermanentEmployee(int employeeNumber1, int employeeType1, string name1, string surname1, string title1, double salaryCoeffi
                      const string &birthDate1, const string &dateOfAppointment, int lengthService1);
};


#endif //AS2_2_PERMANENTEMPLOYEE_H
```



```
#ifndef AS2_2_TEMPORARYEMPLOYEE_H
#define AS2_2_TEMPORARYEMPLOYEE_H
#include "Employee.h"

class TemporaryEmployee: public Employee {
public:
    TemporaryEmployee(int employeeNumber, int employeeType, string name, string surname, string title, double salaryCoefficien
                      const string &birthDate, const string &dateOfAppointment, int lengthService);
}
;

#endif //AS2_2_TEMPORARYEMPLOYEE_H
```

3-6) User Catalog:

If the user chooses “1” or “2”, get information from the user (employee number, employee type, name, surname, salary coefficient, birth date appointment date (service day only for “2”)) add elements in the system.

If the user chooses “3” get the employee number from the user. If the employee doesn’t exist print the main menu again. If exist, update the information of the employee.

If the user chooses “4” get the employee number from user. If employee doesn’t exist print the main menu again. If exist, delete the employee from the system.

If the user chooses “5” get the employee number from the user. If the employee doesn’t exist print the main menu again. If exist, print the information of the employee.

If the user chooses “6” Print all employees' information. If there is no element in the system, print the main menu.

If the user chooses “7” sort the circular-linked-list according to employee appointment date from old to new. Then print all employees. If there is no element in the system, print the main menu.

If the user chooses “8” get the date from the user. If there are employees after the appointment this date print them these employees by appointment date new to old.

If the user chooses “9” get the year from the user. If there are employees who are assigned this year, print these employees' information by appointment date old to new.

If the user chooses “10” get the date from the user. If there are employees who are born before the date, print these employees ascending order by employee number.

If the user chooses “11” get the month from the user. If there are employees who are born this month, print these employees ascending order by employee number.

If the user chooses “12” get the title from the user. If there are employees with this title, print the employee who is last assigned.

If the user chooses another number, the program will print the main menu.

If the program print immediately the main menu, there is no valid employee. For example in “6” if there is no employee in the system print the main menu immediately.

When you add an employee in this system and the employee number exists in this system print the main menu.

If you are trying to delete an employee who does not exist, the main menu will be printed.

If the employe number starts with 0 or birth date or appointment date starts with zero (0d/0m/yyyy) these zeros will not be printed on the board. For example employe number is 0001, it will be printed on the console as 1. For dates 1-1-2000

4) Result, Discussion and Conclusion:

I used a singly circular linked list with array implementation and a doubly-linked list for this assignment. This assignment teaches me especially array implementation of linked lists, teaches me to sort a linked list. I think it is the best assignment to teach these topics. I don’t think my program is efficient when I merge lists. However, I did my best to make efficient this program. The sorting algorithm is the next term topic so that I think my grade is 100 for this assignment.

5) References:

1) https://stackoverflow.com/questions/19579587/sorting-linked-list-c-with-pointers

2) https://www.tutorialspoint.com/cplusplus/relational_operators_overloading.htm

3) Lecture notes