COS10007 Developing Technical Software TP 2 2022

Assignment 1 report

Name: *Hai Nam Ngo*

Student ID: *103488515*

Lab class:

1. *Tuesday / 8.30am-10.30am / TA110*
2. *Friday / 1.30pm-3.30pm / TC223*

Teacher: *Prince Kurumthodathu Surendran*

Due Date: Monday 03rd April 2023 at 11:59 pm

Date Submitted: Monday 03rd April 2023 at 8:44 pm

Question 1

1. Program description

In this question, I need to give out an output is five letters in my name in alphabetical order.

My name is Hai Nam Ngo, so the five letters that I need to insert will be: N G O H A respectively.

I need to insert the first letter is N, second is G, third is O, fourth is H and the last is A, and then the output must be A -> G -> H -> N -> O -> NULL.

2. Inputs and Outputs

The inputs and outputs for this program are described in Table 1.

Table 1. Data dictionary:

Data to be stored	Sample data	Type of data	C type	Input method	In / Out	Variable name
First letter	"N"	text: 1 character	char	manually	printf	ptr1
Second letter	"G"					ptr2
Third letter	"O"					ptr3
Fourth letter	"H"					ptr4
Fifth letter	"A"					ptr5

3. Algorithm

Program steps:

Start

1. Declare a struct called studentname (a node which contains a letter and pointer to the next node)

2. Create displayList function
3. Print “Five letters in my name in alphabetical order”.
4. Make currentPtr = startPtr
5. (While current pointer is not NULL)
   {
   a. Print each letter to the screen
   b. Change the pointer to the next node
   }
6. Print “NULL”

7. In the main function:
8. Declare 5 pointers from ptr1 to ptr5, startPtr and currentPtr.

9. Create a node for the ptr1
10. Insert the value “N” for the ptr1
11. Insert the pointer to next node is NULL
12. Insert the ptr1 as startPtr

13. Create a node for the ptr2
14. Insert the value “G” for the ptr2
15. Insert the pointer to next node is ptr1
16. Insert the ptr2 as startPtr

17. Create a node for the ptr3
18. Insert the value “O” for the ptr3
19. Insert the pointer to next node is NULL
20. Insert the ptr3 as startPtr

21. Create a node for the ptr4
22. Insert the value “H” for the ptr4
23. Insert the pointer to next node is ptr1
24. Insert the ptr1 next node is ptr4
25. Insert the ptr4 as startPtr

26. Create a node for the ptr5
27. Insert the value “A” for the ptr5
28. Insert the pointer to next node is ptr2
29. Insert the ptr5 as startPtr

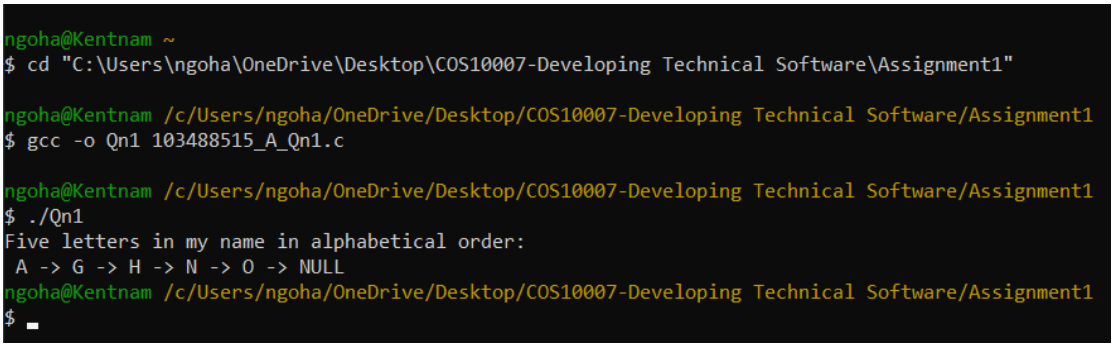
30. Call function displayList
    End

1. Source code:

```
/*
Unit Code: COS10007
Unit Name: Developing Technical Software
Student Name: Hai Nam Ngo
Student ID: 103488515
Name of the file: Ass1_Qn1.c
Brief explanation: print out five letters from my name in alphabetical order
Input: No input required
Output: A -> G -> H -> N -> O -> NULL (Hai Nam Ngo -> N G O H A  -> A G H N
O)
Date created: 3/22/2023
Last modified: 3/28/2023
*/

#include <stdio.h>
#include <stdlib.h>


/*structure listnode*/
struct studentname
{
	char letter; /* each listNode contains a character from the student name*/
	struct studentname *next; /*pointer to the next node*/
};/*end listnode*/


typedef struct studentname STUDENTName; /* synonym for struct studentname*/
typedef STUDENTName *STUDENTNamePtr; /* synonym for studentname* */

//prototype
void displayList(STUDENTNamePtr currentPtr,STUDENTNamePtr startPtr);

//function for print out the output
void displayList(STUDENTNamePtr currentPtr,STUDENTNamePtr startPtr)
{
	printf("Five letters in my name in alphabetical order: \n");
	currentPtr=startPtr;
	while(currentPtr!=NULL)
	{
		printf(" %c ->",currentPtr->letter);
		currentPtr=currentPtr->next;
	}
	printf(" NULL");
}

/*start main*/
int main(void)
{
	STUDENTNamePtr ptr1;
```

```
        STUDENTNamePtr ptr2;
        STUDENTNamePtr ptr3;
        STUDENTNamePtr ptr4;
        STUDENTNamePtr ptr5;
        STUDENTNamePtr startPtr;
        STUDENTNamePtr currentPtr;

        /*Insert the first letter: N*/
        ptr1=(STUDENTName*)malloc(sizeof(STUDENTName));
        ptr1->letter='N';
        ptr1->next=NULL;
        startPtr=ptr1;

        /*Insert the second letter: G...should go before N*/
        ptr2=(STUDENTName*)malloc(sizeof(STUDENTName));
        ptr2->letter='G';
        ptr2->next=ptr1;
        startPtr=ptr2;

        /*Insert the third letter: O...should go after N*/
        ptr3=(STUDENTName*)malloc(sizeof(STUDENTName));
        ptr3->letter='O';
        ptr3->next=NULL;
        ptr1->next=ptr3; //this means the next node for N will be O
        startPtr=ptr3;

        /*Insert the fourth letter: H...should go between G and N*/
        ptr4=(STUDENTName*)malloc(sizeof(STUDENTName));
        ptr4->letter='H';
        ptr4->next=ptr1;
        ptr2->next=ptr4;
        startPtr=ptr4;

        /*Insert the fifth letter: A...should go before G*/
        ptr5=(STUDENTName*)malloc(sizeof(STUDENTName));
        ptr5->letter='A';
        ptr5->next=ptr2;
        startPtr=ptr5;

        displayList(currentPtr,startPtr);
}
```

2. **Screenshots showing working program (Show all possible outcome):**



Question 2

1. Program description

Create an Employee Management System with 8 functions including the menu, display list of Employee, search for one Employee, find the Employee which has the largest salary, display list of employees which have the salary below 5000, find the average salary of a company, insert a new employee to the list and quit the program.

2. Inputs and Outputs

The inputs and outputs for this program are described in Table 1.

Table 1. Data dictionary:

Data to be stored	Sample data	Type of data	C type	Input method	In / Out	Variable name
option	“1”,”2”,”3”,…	integer	int	scanf	function	option
name	“Kent”	string	Char[20]	fscanf scanf	printf	name
age	“30”	Integer	Int	fscanf scanf	printf	age
Id	“1008”	Integer	Int	fscanf scanf	printf	deptId
company	“Apple”	String	Char[20]	fscanf scanf	printf	cmpName
salary	“4000.00000”	float	float	fscanf scanf	printf	salary
User input to search	“Kent”	string	Char[20]	scanf		search

3. Algorithm

Program steps:

3. Start the program.
4. Define a self-referential structure for employee details that includes personal and official information.
5. Create a function named readFile to read the data from the "employee.txt" file and store it in a linked list in alphabetical order.
6. Create a function named displayEmployees to display all employees' information to the screen in alphabetical order.
7. Create a function named searchEmployee to search for an employee's information based on their name.
8. Create a function named findMaximum to find the employee with the highest salary and print their information.
9. In the main function, call the readFile function to read the employee data from the file.
10. Display a menu of options for the user to choose from: display all employees, search for an employee, find the employee with the highest salary, or exit the program.
11. Based on the user's choice, call the appropriate function to perform the action.
12. Repeat step 8 and 9 until the user chooses to exit the program.
13. End the program.

4. Source code:

```
/*
Unit Code: COS10007
Unit Name: Developing Technical Software
Student Name: Hai Nam Ngo
Student ID: 103488515
Name of the file: Ass1_Qn2.c
Brief explanation: Create an Employee Management System with 8 functions
including the menu
Input: Name, age, id, comany, salary, search
Output: Display the menu and 8 functions.
Date created: 3/22/2023
Last modified: 4/02/2023
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

//self-referential structure
struct personTag
{
	char name[20];
	int age;
};

struct officialTag
```

```
{
        int deptId;
        char cmpName[20];
        double salary;
};

struct employeeTag
{
        struct personTag personalInfo;
        struct officialTag officialInfo;
        struct employeeTag *next;
};

typedef struct employeeTag EmployeeTag;
typedef EmployeeTag *EmployeeTagPtr;


//readfile function: to open the employee.txt and read the details and save it in
linkedlist
EmployeeTagPtr readFile()
{
   FILE *fp=fopen("employee.txt","r");
   if(fp==NULL)
   {
      printf("Error opening the file \n");
      return NULL;
   }

   EmployeeTagPtr startPtr=NULL;
   EmployeeTagPtr currentPtr=NULL;
   EmployeeTagPtr newPtr=NULL;

   // Read the first employee from the file
   newPtr = (EmployeeTag*)malloc(sizeof(EmployeeTag));
   fscanf(fp,"%s",newPtr->personalInfo.name);
   fscanf(fp,"%d",&newPtr->personalInfo.age);
   fscanf(fp,"%d",&newPtr->officialInfo.deptId);
   fscanf(fp,"%s",newPtr->officialInfo.cmpName);
   fscanf(fp,"%lf",&newPtr->officialInfo.salary);
   newPtr->next = NULL;
   startPtr = newPtr;
   currentPtr = newPtr;

   // Read the remaining employees from the file
   while (!feof(fp))
   {
      newPtr = (EmployeeTag*)malloc(sizeof(EmployeeTag));
      fscanf(fp,"%s",newPtr->personalInfo.name);
      fscanf(fp,"%d",&newPtr->personalInfo.age);
      fscanf(fp,"%d",&newPtr->officialInfo.deptId);
```

```
        fscanf(fp,"%s",newPtr->officialInfo.cmpName);
        fscanf(fp,"%lf",&newPtr->officialInfo.salary);
        newPtr->next = NULL;

        // Insert the new employee in alphabetical order based on name
        if (strcmp(newPtr->personalInfo.name, startPtr->personalInfo.name) < 0)
        {
           newPtr->next = startPtr;
           startPtr = newPtr;
        }
        else
        {
           currentPtr = startPtr;
           while (currentPtr->next != NULL && strcmp(newPtr->personalInfo.name,
currentPtr->next->personalInfo.name) > 0)
           {
              currentPtr = currentPtr->next;
           }
           newPtr->next = currentPtr->next;
           currentPtr->next = newPtr;
        }
     }

     // Close the file
     fclose(fp);
     return startPtr;
}


//this function is used to display employee list to the screen
void displayEmployees(EmployeeTagPtr startPtr)
{
        printf("-------------------------------------------------------------------------\n");
        printf("\t\tList of all employees (alphabetical order)\n");
        printf("Name\tAge\tID\tCompany\tSalary \n");

        // loop through the list and print each node
        while (startPtr != NULL)
        {
                printf("%s\t%d\t%d\t%s\t%lf \n",
                startPtr->personalInfo.name,
                startPtr->personalInfo.age,
                startPtr->officialInfo.deptId,
                startPtr->officialInfo.cmpName,
                startPtr->officialInfo.salary);

                startPtr = startPtr->next;
        }
}
```

```
//this function is used to search for specific employee in the list
void searchEmployee(EmployeeTagPtr startPtr)
{
	//ask the user to type in the name of the employee
   char search[20];
   printf("-------------------------------------------------------------------------\n");
   printf("Enter the name of the employee: \n");
   scanf("%s", search);

	//this while loop is created to run through every pointers in the list
   while (startPtr != NULL)
   {
		//this command is created to compare if the user's input is the same as
the data in the pointer
      if (strcmp(search, startPtr->personalInfo.name) == 0)
      {
			//print out the information
			printf("-------------------------------------------------------------------
----\n");
         printf("Name\tAge\tID\tCompany\tSalary \n");
         printf("%s\t%d\t%d\t%s\t%lf \n",
			startPtr->personalInfo.name,
			startPtr->personalInfo.age,
			startPtr->officialInfo.deptId,
			startPtr->officialInfo.cmpName,
			startPtr->officialInfo.salary);
         break;
      }
		//if it's not the same, the system will continue to find
      else
      {
         startPtr = startPtr->next;
      }
   }
	//if the user's input doesn't match with any name in the list
   if (startPtr == NULL)
   {
      printf("No result found\n");
   }
}

//this function is created to find the employee has the largest salary
void findMaximum(EmployeeTagPtr startPtr)
{
	//declare a new pointer called maxEmployee
   double maximum = startPtr->officialInfo.salary;
   EmployeeTagPtr maxEmployee = startPtr;

	//while loop to run through the list
   while (startPtr != NULL)
```

```
    {
                //for each node, if the salary is higher than the "maximum"
        if (startPtr->officialInfo.salary > maximum)
        {
                        //update the value of maximum equal to the salary of that
employee
            maximum = startPtr->officialInfo.salary;
            maxEmployee = startPtr;
        }
                //run to the next node
        startPtr = startPtr->next;
    }
            //print out the information
    printf("--------------------------------------------------------------------------\n");
            printf("The employee with the largest salary: \n");
    printf("Name\tAge\tID\tCompany\tSalary \n");
    printf("%s\t%d\t%d\t%s\t%lf \n",
            maxEmployee->personalInfo.name,
            maxEmployee->personalInfo.age,
            maxEmployee->officialInfo.deptId,
            maxEmployee->officialInfo.cmpName,
            maxEmployee->officialInfo.salary);
}

//this function is created to display the list of employee who has salary below 5000
void lowerSalary(EmployeeTagPtr startPtr)
{
            //created a node called "lowEmployee"
    EmployeeTagPtr lowEmployee = startPtr;
    int found = 0;
    printf("--------------------------------------------------------------------------\n");
    printf("Employees with salary less than 5000: \n");
    printf("Name\tAge\tID\tCompany\tSalary \n");

            //while loop run through the list
    while (lowEmployee != NULL)
    {
                //if the salary of the employee is smaller than 5000
        if (lowEmployee->officialInfo.salary < 5000)
        {
                        //print out the information of that employee
            printf("%s\t%d\t%d\t%s\t%lf \n",
                lowEmployee->personalInfo.name,
                lowEmployee->personalInfo.age,
                lowEmployee->officialInfo.deptId,
                lowEmployee->officialInfo.cmpName,
                lowEmployee->officialInfo.salary);
                        //this means the statement is true
            found = 1;
        }
```

```
                //go to the next node
        lowEmployee = lowEmployee->next;
    }
        //if there is no employee with salary less than 5000
        //this means found=0
    if (!found)
    {
        printf("No employees with salary less than 5000 found\n");
    }
}

//this function is created to caculate the average salary of each company
void averageSalary(EmployeeTagPtr startPtr)
{
    char company[20];
    int count = 0;
    double totalSalary = 0;
    EmployeeTagPtr currentPtr = startPtr;

        //ask user for the name of the company
    printf("--------------------------------------------------------------------------\n");
    printf("Enter a company name (Apple, Samsung, Oracle): ");
    scanf("%s", company);

        //run through the file
    while (currentPtr != NULL)
    {
                //if the user's input is the same as the data in the file
        if (strcmp(currentPtr->officialInfo.cmpName, company) == 0)
        {
                        //update the total salary (this is also known as the sum of
employees' salary of that company)
            totalSalary += currentPtr->officialInfo.salary;
                        //update the number of the employee
            count++;
        }
        //go to the next node
        currentPtr = currentPtr->next;
    }
        //if count>0, means that it has data to be displayed
    if (count > 0)
    {
                //display the output
        double average = totalSalary / count;
        printf("--------------------------------------------------------------------------\n");
        printf("The average salary of %s Company is: %lf\n", company, average);
    }
        //if there is no company name like the user's input
    else
    {
```

```
        printf("--------------------------------------------------------------------------\n");
        printf("No company found.\n");
    }
}

//this function is created to update file: insert new employee to the list
void updateFile(EmployeeTagPtr startPtr)
{
    // open the file in append mode
    FILE *fp = fopen("employee.txt", "a");
    if (fp == NULL)
            {
        printf("Error opening file.\n");
        return;
    }

    // prompt the user for new employee's details
    EmployeeTagPtr newEmployee = malloc(sizeof(EmployeeTag));

    printf("Enter new employee details:\n");
    printf("ID: ");
    scanf("%d", &newEmployee->officialInfo.deptId);
    printf("Name: ");
    scanf("%s", newEmployee->personalInfo.name);
            printf("Age: ");
            scanf("%d",&newEmployee->personalInfo.age);
    printf("Company name: ");
    scanf("%s", newEmployee->officialInfo.cmpName);
    printf("Salary: ");
    scanf("%lf", &newEmployee->officialInfo.salary);

    // write the new data at the end of the file
    fprintf(fp, "%s\t%d\t%d\t%s\t%lf\n",
        newEmployee->personalInfo.name,
                    newEmployee->personalInfo.age,
        newEmployee->officialInfo.deptId,
        newEmployee->officialInfo.cmpName,
        newEmployee->officialInfo.salary);

    // close the file
    fclose(fp);

    // add the new employee to the LinkedList
    // find the last node in the LinkedList
    EmployeeTagPtr currentPtr = startPtr;
    while (currentPtr->next != NULL)
            {
        currentPtr = currentPtr->next;
    }
    // add the new node to the end of the LinkedList
```

```
   currentPtr->next = newEmployee;
   newEmployee->next = NULL;

        printf("---------------------------------------------------------------------------\n");
        printf("Update File Completed.\n");
}

//this function is created to display the instruction to the user
int menu()
{
   int option;

        //display the menu
        printf("---------------------------------------------------------------------------\n");
   printf("(1) Display employee's details \n");
   printf("(2) Search for an employee's salary \n");
   printf("(3) Find the details of employee with the largest salary \n");
   printf("(4) Find the details of all employees having salary less than 5000 \n");
   printf("(5) Find the average salary of a company \n");
   printf("(6) Add new employee to the record \n");
   printf("(7) Quit program \n");
   printf("Select your option: ");
   scanf("%d", &option); // add this line to read the selected option

   return option; // add this line to return the selected option to the main function
}

//main function to link all of the functions together
int main()
{
   int option;

        //print out welcome message
   printf("---------------------------------------------------------------------------\n");
   printf("\t\tWelcome to The Employee Management System \n");
   EmployeeTagPtr startPtr = readFile();

   do
   {
                //option is taken from the menu function
      option = menu();
      switch(option)
      {
         case 1:
            displayEmployees(startPtr);
            break;

         case 2:
            searchEmployee(startPtr);
            break;
```

```
            case 3:
                                findMaximum(startPtr);
                break;

            case 4:
                                lowerSalary(startPtr);
                break;

            case 5:
                                averageSalary(startPtr);
                break;

            case 6:
                                updateFile(startPtr);
                break;

            case 7:
                                printf("------------------------------------------------------------
--------\n");
                printf("Thank you for using the Employee Management System. Exiting the
program.\n");
                                printf("------------------------------------------------------------
--------------\n");
                break;

            default:
                printf("Invalid option selected. Please try again.\n");
                break;
        }
    } while (option != 7);

    return 0;
}
```

5. Screenshots showing working program (Show all possible outcome):

```
ngoha@Kentnam /c/Users/ngoha/OneDrive/Desktop/COS10007-Developing Technical Software/Assignment1
$ gcc -o Qn2 103488515_A_Qn2.c

ngoha@Kentnam /c/Users/ngoha/OneDrive/Desktop/COS10007-Developing Technical Software/Assignment1
$ ./Qn2
--------------------------------------------------------------------------
               Welcome to The Employee Management System
--------------------------------------------------------------------------
(1) Display employee's details
(2) Search for an employee's salary
(3) Find the details of employee with the largest salary
(4) Find the details of all employees having salary less than 5000
(5) Find the average salary of a company
(6) Add new employee to the record
(7) Quit program
Select your option: 1
--------------------------------------------------------------------------
               List of all employees (alphabetical order)
Name    Age     ID      Company Salary
Bach    19      1010    Oracle  8000.000000
Billy   21      1009    Samsung 5000.000000
Joseph  50      1002    Oracle  4000.000000
Kent    20      1006    Apple   10000.000000
Lilly   40      1004    Samsung 7000.000000
Mary    40      1003    Samsung 2000.000000
Nathan  26      1007    Oracle  9000.000000
Paul    20      1008    Samsung 6000.000000
Peter   30      1001    Apple   1000.000000
Tony    50      1005    Oracle  3000.000000
```

```
--------------------------------------------------------------------------
(1) Display employee's details
(2) Search for an employee's salary
(3) Find the details of employee with the largest salary
(4) Find the details of all employees having salary less than 5000
(5) Find the average salary of a company
(6) Add new employee to the record
(7) Quit program
Select your option: 2
--------------------------------------------------------------------------
Enter the name of the employee:
Lilly
--------------------------------------------------------------------------
Name    Age     ID      Company Salary
Lilly   40      1004    Samsung 7000.000000
--------------------------------------------------------------------------
(1) Display employee's details
(2) Search for an employee's salary
(3) Find the details of employee with the largest salary
(4) Find the details of all employees having salary less than 5000
(5) Find the average salary of a company
(6) Add new employee to the record
(7) Quit program
Select your option:
```

```
--------------------------------------------------------------------------
(1) Display employee's details
(2) Search for an employee's salary
(3) Find the details of employee with the largest salary
(4) Find the details of all employees having salary less than 5000
(5) Find the average salary of a company
(6) Add new employee to the record
(7) Quit program
Select your option: 2
--------------------------------------------------------------------------
Enter the name of the employee:
Test
No result found
```

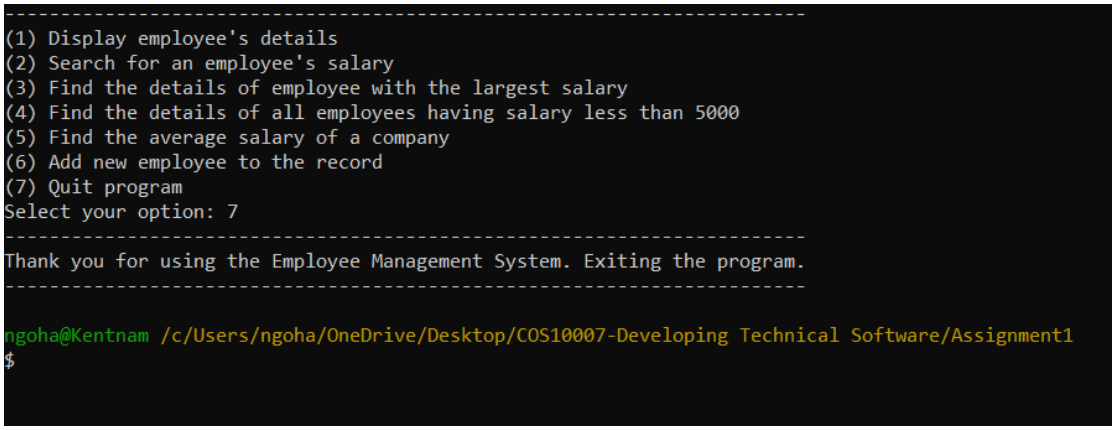
```
--------------------------------------------------------------------------
(1) Display employee's details
(2) Search for an employee's salary
(3) Find the details of employee with the largest salary
(4) Find the details of all employees having salary less than 5000
(5) Find the average salary of a company
(6) Add new employee to the record
(7) Quit program
Select your option: 3
--------------------------------------------------------------------------
The employee with the largest salary:
Name    Age     ID      Company Salary
Kent    20      1006    Apple   10000.000000
--------------------------------------------------------------------------
(1) Display employee's details
(2) Search for an employee's salary
(3) Find the details of employee with the largest salary
(4) Find the details of all employees having salary less than 5000
(5) Find the average salary of a company
(6) Add new employee to the record
(7) Quit program
Select your option:
```

```
--------------------------------------------------------------------------
(1) Display employee's details
(2) Search for an employee's salary
(3) Find the details of employee with the largest salary
(4) Find the details of all employees having salary less than 5000
(5) Find the average salary of a company
(6) Add new employee to the record
(7) Quit program
Select your option: 4
--------------------------------------------------------------------------
Employees with salary less than 5000:
Name    Age     ID      Company Salary
Joseph  50      1002    Oracle  4000.000000
Mary    40      1003    Samsung 2000.000000
Peter   30      1001    Apple   1000.000000
Tony    50      1005    Oracle  3000.000000
--------------------------------------------------------------------------
(1) Display employee's details
(2) Search for an employee's salary
(3) Find the details of employee with the largest salary
(4) Find the details of all employees having salary less than 5000
(5) Find the average salary of a company
(6) Add new employee to the record
(7) Quit program
Select your option:
```

```
--------------------------------------------------------------------------
(1) Display employee's details
(2) Search for an employee's salary
(3) Find the details of employee with the largest salary
(4) Find the details of all employees having salary less than 5000
(5) Find the average salary of a company
(6) Add new employee to the record
(7) Quit program
Select your option: 5
--------------------------------------------------------------------------
Enter a company name (Apple, Samsung, Oracle): Apple
--------------------------------------------------------------------------
The average salary of Apple Company is: 5500.000000
--------------------------------------------------------------------------
(1) Display employee's details
(2) Search for an employee's salary
(3) Find the details of employee with the largest salary
(4) Find the details of all employees having salary less than 5000
(5) Find the average salary of a company
(6) Add new employee to the record
(7) Quit program
Select your option:
```

```
--------------------------------------------------------------------------
(1) Display employee's details
(2) Search for an employee's salary
(3) Find the details of employee with the largest salary
(4) Find the details of all employees having salary less than 5000
(5) Find the average salary of a company
(6) Add new employee to the record
(7) Quit program
Select your option: 5
--------------------------------------------------------------------------
Enter a company name (Apple, Samsung, Oracle): None
--------------------------------------------------------------------------
No company found.
--------------------------------------------------------------------------
```

```
(1) Display employee's details
(2) Search for an employee's salary
(3) Find the details of employee with the largest salary
(4) Find the details of all employees having salary less than 5000
(5) Find the average salary of a company
(6) Add new employee to the record
(7) Quit program
Select your option: 6
Enter new employee details:
ID: 1234
Name: Test
Age: 30
Company name: Asus
Salary: 1234
--------------------------------------------------------------------------
Update File Completed.
--------------------------------------------------------------------------
(1) Display employee's details
(2) Search for an employee's salary
(3) Find the details of employee with the largest salary
(4) Find the details of all employees having salary less than 5000
(5) Find the average salary of a company
(6) Add new employee to the record
(7) Quit program
Select your option: 1
--------------------------------------------------------------------------
                List of all employees (alphabetical order)
Name    Age     ID      Company Salary
Bach    19      1010    Oracle  8000.000000
Billy   21      1009    Samsung 5000.000000
Joseph  50      1002    Oracle  4000.000000
Kent    20      1006    Apple   10000.000000
Lilly   40      1004    Samsung 7000.000000
Mary    40      1003    Samsung 2000.000000
Nathan  26      1007    Oracle  9000.000000
Paul    20      1008    Samsung 6000.000000
Peter   30      1001    Apple   1000.000000
Tony    50      1005    Oracle  3000.000000
Test    30      1234    Asus    1234.000000
--------------------------------------------------------------------------
```



Question 3

1. Program description

This code create a train system that schedules trains, by using structure.
There are 6 functions in the code:
First function: auto generate ID for the train
Second function: auto generate the departure time for the train.
Third function: display the departure list to the screen.
Fourth function: Reshedule the list from earliest to latest.
Fifth function: allow the first train in the stack to leave (the earliest)
Sixth function: allow the last train in the stack to leave (emergency)

Then use the main to call every function

2. Inputs and Outputs

The inputs and outputs for this program are described in Table 1.

Table 1. Data dictionary:

Data to be stored	Sample data	Type of data	C type	Input method	In / Out	Variable name
Train Id	“245”	Integer	int	auto	printf	id
schedule	03 21:09:17 2023	time	Time_t	auto	printf	schedule

3. Algorithm

Program steps:

1. Create a struct Train that has an integer ID, a time_t schedule, and a pointer to the next train
2. Define a function called givetrainid that takes in an array of trains and its size as parameters
3. Inside the function, seed the random number generator with the current time
4. For each train in the array, generate a random ID between 100 and 200 and set its schedule to 0

5. Define a function called givedepartureтime that takes in an array of trains and its size as parameters
6. Inside the function, for each train in the array, set its schedule to the current time plus a random number of seconds between 0 and 9999
7. Define a function called displayschedule that takes in an array of trains and its size as parameters
8. Inside the function, print out the train ID and its departure time to the screen
9. Define a function called reschedule that takes in an array of trains and its size as parameters
10. Inside the function, use bubble sort to sort the trains in the array by their departure time, from earliest to latest
11. Define a function called LeaveEarliestTrain that takes in an array of trains and a pointer to its size as parameters
12. Inside the function, if the size of the array is 0, print out a message saying there are no trains left in the station and return
13. Otherwise, print out the ID and departure time of the train with the earliest schedule to the screen and pop it out of the array
14. Define a function called EmergencyLeave that takes in an array of trains and a pointer to its size as parameters
15. Inside the function, if the size of the array is 0, print out a message saying there are no trains left in the station and return
16. Otherwise, print out the ID and departure time of the train with the latest schedule to the screen and pop it out of the array

    In the main function:
17. Create an array of trains with size 10
18. Call givetrainid to generate random IDs for the trains
19. Call givedepartureтime to assign random departure times for the trains
20. Print out the initial schedule using displayschedule
21. Sort the trains by their departure time using reschedule
22. Print out the final schedule using displayschedule
23. Call LeaveEarliestTrain to make the train with the earliest departure time leave the station
24. Call EmergencyLeave to make the train with the latest departure time leave the station
25. Print out the final schedule again using displayschedule

4. Source code:

```
/*
Unit Code: COS10007
Unit Name: Developing Technical Software
Student Name: Hai Nam Ngo
Student ID: 103488515
Name of the file: 103488515_A_Qn3.c
Brief explanation: This code create a train system that schedules trains, by using
structure. There are 6 functions in the code.
Input: No input needed
Output: Train schedule list
Date created: 3/22/2023
Last modified: 4/03/2023
```

```
*/

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

struct Train
{
   int id;
   time_t schedule;
};


// This function generates random train IDs from 100 to 200 and assigns them to the corresponding trains in the array
void givetrainid(struct Train trains[], int size)
{
   // Seed the random number generator with the current time
   srand(time(NULL));

   // For each train in the array, generate a random ID and set its schedule to 0
   for (int i = 0; i < size; i++)
   {
      trains[i].id = rand() % 101 + 100;  // Generate a random integer between 0 and 100, and add 100 to get a random integer between 100 and 200
      trains[i].schedule = 0;  // Set the train's schedule to 0
   }
}

// This function assigns a random departure time to each train in the array
void givedeparturetime(struct Train trains[], int size)
{
   // For each train in the array, set its schedule to the current time plus a random number of seconds between 0 and 9999
   for (int i = 0; i < size; i++)
   {
      trains[i].schedule = time(NULL) + rand() % 9999;
   }
}

//display the schedule to the screen
void displayschedule(struct Train trains[], int size)
{
   printf("Train\tTime\n");
   for (int i = 0; i < size; i++)
   {
      printf("%d\t%s", trains[i].id, ctime(&trains[i].schedule));
   }
}
```

```
//display the schedule list in order (from the earliest to the latest)
void reschedule(struct Train trains[], int size)
{
   struct Train swap;
   //going through every train except the last one, because we want to compare the
time of two train (two node next to each other)
   for (int i = 0; i < size - 1; i++)
   {
      //going from the second train two the last train
      for (int j = i + 1; j < size; j++)
      {
         //if the departure time of the next train is earlier than the current train, swap
their positions
         if (trains[i].schedule > trains[j].schedule)
         {
            //swap place without losing its time value
            swap = trains[i];
            trains[i] = trains[j];
            trains[j] = swap;
         }
      }
   }
}

//the function is created to make the train with the earliest schedule leave the station
void LeaveEarliestTrain(struct Train trains[], int *size)
{
         int i=0;
         //if size=0, means the stack have no data (no trains)
   if (*size == 0)
         {
                  //print out the information to the screen
      printf("There are no trains left in the station.\n");
      return;
   }

         //print out the display to the screen
         printf("-----------------------------------------------------\n");
         printf("\t\tAnnouncement\n");
   printf("Train ID %d\nSchedule: %s \nThis train is leaving the station.\n",
trains[i].id, ctime(&trains[i].schedule));
         //pop the first train out of the stack
   for (i = 0; i < *size - 1; i++)
         {
      trains[i] = trains[i + 1];
   }
   (*size)--;
}
```

```
////the function is created to make the train with the latest schedule leave the station
(emergency)
void EmergencyLeave(struct Train trains[], int *size)
{
        //if size=0, means the stack have no data (no trains)
   if (*size == 0)
        {
                //print out the information to the screen
      printf("There are no trains left in the station.\n");
      return;
   }
        //print out the display to the screen
        printf("-------------------------------------------------\n");
        printf("\t\tEmergency Alert\n");
   printf("Train ID %d\nSchedule: %s\nThis train is leaving the station NOW.\n",
trains[*size - 1].id, ctime(&trains[*size - 1].schedule));
   (*size)--;
}


int main()
{
   int size = 10;
   struct Train trains[size];

   // Step 1
   givetrainid(trains, size);

   // Step 2
   givedeparturetime(trains, size);

   // Step 3
   printf("-------------------------------------------------\n");
   printf("\t\tInitial schedule\n");
   printf("-------------------------------------------------\n");
   displayschedule(trains, size);

   // Step 4
   reschedule(trains, size);

   // Step 3 (repeated)
   printf("-------------------------------------------------\n");
   printf("\t\tFinal schedule\n");
   printf("-------------------------------------------------\n");
   displayschedule(trains, size);

   // Step 5
   LeaveEarliestTrain(trains, &size);

   //Step 6
```

```
    EmergencyLeave(trains, &size);

        // Step 3 (repeated)
        printf("--------------------------------------------------\n");
    printf("\tFinal schedule (after two train left)\n");
    printf("--------------------------------------------------\n");
    displayschedule(trains, size);

    return 0;
}
```

5. Screenshots showing working program (Show all possible outcome):