COS10007 - Developing Technical Software
Assignment 1 (25 Marks)
Due date – Check Canvas

You will have to demonstrate/explain/modify your work to your COS10007 teacher, if you are absent/unavailable or fail to demonstrate properly, zero marks will be awarded.

Please note, this is an individual task, and it will be checked for plagiarism. All the involved parties will be penalised if any plagiarism is found.

Please visit https://goo.gl/hQ87zg for more details.

Instructions

1. This assignment contains 3 questions. Q1 is for 5 marks, Q2 is for 10 marks, and Q3 is for 10 marks. The total assignment is for 25 marks and refer to the detailed rubric given on Assignment page in Canvas for mark allocation.
2. Submit one-word document and a zip file. Use the following format to prepare the word document (use the report template available in assignment page).

   a. Question No. (No need to copy and paste question)
   b. Problem description, table, pseudo-code/flowchart etc.
   c. C program - copy and paste your c program (not the screenshot of the code)
   d. **Enough** screenshots of the output that shows **all possible outcome**.
3. Marks will be given for proper **indentation and comments.**
4. **Assignment Demonstration** is mandatory.

Other requirements:

- This assignment must be written in c.
- Your code must have appropriate header (multiline/block) comments including your name and student number, the name of the .c file, the purpose of the program, brief explanations of variables and explanations of any code, which is not obvious to another programmer, summarising the input, output and local variables as well as expressions used in your program and test data.
- Include inline (single line) comments throughout the program describing important statements.
- Use appropriate and descriptive variable following the naming rules and conventions.
- Write a brief (no more than several pages) report, which illustrates your program design (algorithm or flowchart, identification of variables, constants) and include evidence of testing – screen shots or pasted output text of several tests, and the contents of the .c file.
- Marks will be allocated depending on the amount of original work submitted. Marks will be deducted for plagiarised and/or un-attributed work.

Question 1 (5 Marks)

Create a linked list using the following structure

```
struct studentname {
char letter;
struct studentname *next;
};

typedef struct studentname STUDENTName;
typedef STUDENTName *STUDENTNamePtr;
```

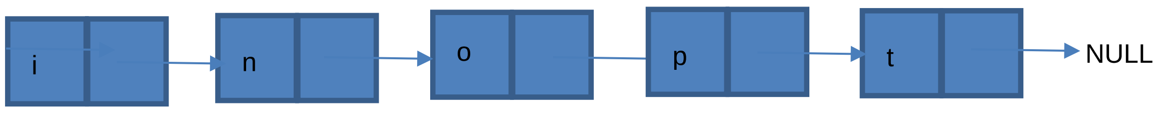
Create a linked list manually (without using any loops or recursive functions or any functions) that contains five nodes where the data part (structure element letter) of the nodes should be the first five letters of your last name. One letter will go to one node and the node insertion should happen one after another in alphabetical order. When you insert a new node, it should be in the right place to get alphabetical order as shown in previous task. If your last name doesn't have five letters fill the remaining nodes with letters from your first name starting from the first letter (Eg: if your name is Devin Ly, then nodes will contain L, y, D, e and v).

Example:

Assume your name is Ricky Ponting; take the first five letters of the last name, which is Ponti so the insertion order is

```
newptr = new STUDENTName;
newptr -> letter = 'P ';
                    .
                            .
                    o
                    .
                    .
                    n
                    .
                    t
                    .
                    i
```

and when you print the linked list it should be like as shown below (Note, you are not using any sorting to get this in alphabetical order, you manually insert nodes in the right place to get it in alphabetical order).



Then write a *displayList* function to print the nodes on the screen.

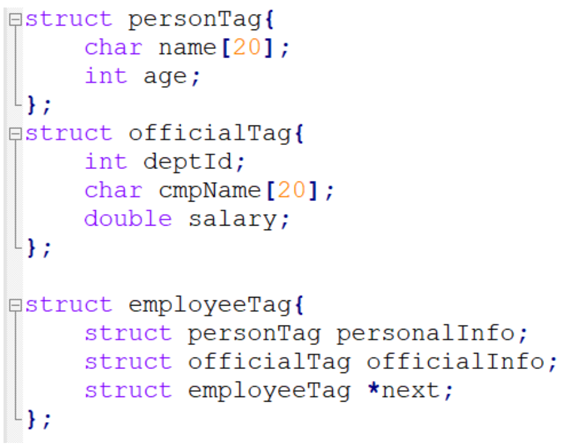
Question 2 (10 Marks)

A database of details of employees is kept in a text file as shown below. The entries are employee name, age, department id, company name and salary in that order. A small segment of the file might look like the following.

Name	Age	id	Company	Salary
Peter	30	1001	Apple	8000
Joseph	50	1002	Oracle	4000
Mary	40	1003	Samsung	6000
Lilly	40	1203	Samsung	7000
Tony	50	1002	Oracle	3000

So, according to the data presented in this file, the first employee's name is Peter; his age is 30, his department id is 1001, his company name is Apple and has a salary of 3000. The second employee's name is Joseph; his age is 50, his department id is 1002, his company name is Oracle and has a salary of 4000 and so on.
Step 1: Manually create a text file that contains the employees' details (minimum 10 employees) as shown in the problem, you can enter your own data for the file, but it should follow the same format. Save the text file as ***employee.txt.***

Use the structure given below for this problem.



```
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;
};
```

Create a LinkedList using the above self-referential structure and read the contents of the text file into LinkedList. Each node of the LinkedList contains details of one employee. Implement all functionalities mentioned below:

The main()function handles all interactions with the **user** and other functions:

- It displays an appropriate welcome message introducing the program.
  Calls a function named readFile()which opens a text file employee.txt (a sample text file is shown above) for reading and storing all of the employees' details from the file to a LinkedList in order of name (insertion should happen in alphabetical order). It then repeatedly calls the menu()function to display user options, get the user selection returned by the menu() function, use a switch (or if ..else if) statement to process user request by calling appropriate function(s).
- It displays the result with an appropriate message after processing user request.
- It displays a goodbye message when the user selects the **Quit** option from

the menu and terminates the program.

The menu()function has no parameters. When called, it displays a menu of **6** options allowing the user to select one and returns this option to the calling main()function.
The options displayed should be:
(1) Display employees' 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

- **Option (1)** will use a function called `displayEmployees()` called from the main()to display the contents of the LinkedList on the screen in an appropriate format.
- **Option (2)** will use a function called `searchEmployee`()which is designed to search for an employee. Display all the details of that employee, if no such employee is found, report it back to the user.
- **Option (3)** will use a function called `findMaximum`() which is designed to find the details of employee having the largest salary in the LinkedList. Display all the details of that employee.
- **Option (4)** will use a function called `lowerSalary`()which is to find the employees who have salary less than 5000. Display all the details of these employees.
- **Option (5)** will use a function called `averageSalary()` which is used to find the avarage salary of all emplyees of a company. If there is no such company, the program should report it back to the user.
- **Option (6)** will first use a function called `updateFile`() which will open the same text file in **append** mode, prompt the user for new employee's details and then write the new data at the end of the file using the same format as the original file. It will then the call the `readFile`() function again to read the contents of the updated file and repopulate the LinkedList. Call the display function again to show the employees' details on screen.
- **Option (7)** will terminate the program after displaying an appropriate goodbye message.

Question 3 (10 Marks)

In this problem you are required to create a train system that schedules trains. Create a structure that contains two elements, first element should store train *id*, second element should store schedule (time). Declare an array of this structure with a size 10 and do the following steps.

- Step 1 - from main call a function (give an appropriate name for this function) passing the array (in this problem array means array of the structure) and size of the array as parameters to fill the details of trains in the array. The train *id* should be filled with a random integer between 100 and 200. This function should be called first and need to call only once.
- Step 2 – from main call a function (give an appropriate name for this function) passing the array and size of the array as parameters to fill the departure time of the trains (fill the time field of the structure in the array). You can use the sample code provided to fill the time details. You need to call this function only once and must call only after invoking the function in step 1.

- Step 3 - from main call a function (give an appropriate name for this function) passing the array and size of the array as parameters to print the details of array (details of all trains – train id and time) on the screen. You can call this function repeatedly.
- Step 4 - from main call a function (give an appropriate name for this function) passing the array and size of the array as parameters to schedule the trains based on the time field of the train. The earliest (time value) train to be placed at the bottom of the array. You need to call this function only once.

Then extend your structure to include a self-referential structure element to create queue and stack using this structure. This self-referential structure should be used for the next two functions. These functions can be called any number of times and in any order but can be called only after scheduling the trains (step 4) based on the time field of the trains.

- Step 5 - call a function (give an appropriate name for this function) that allows one train to leave the station based on the schedule. Train with the earliest schedule should leave the station first.
- Step 6 **- c**all a function (give an appropriate name for this function) that allows one train to leave the station with the last scheduled train from the array to leave the station first (in an emergency).

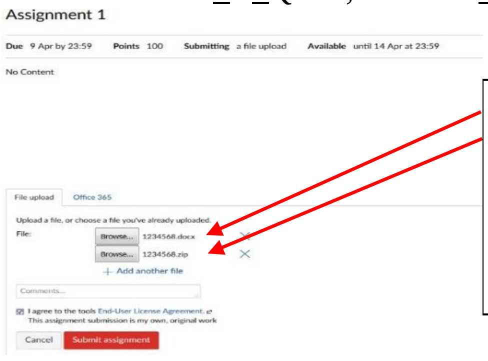
<u>Assignment submission information:</u>
Submissions through **Canvas** must be made on or before the due date/time.
Each submission should have two files.
1. A report (name of the report should be with your student number, eg: 1012546_assignment1.docx) – use template provided with this assignment.

This report will be used for plagiarism check using turnitin software. **20% of marks will be deducted if this report is missing for plagiarism check.** Report must (***.doc/docx)*** contain:

- Description of the problem
- A copy of the contents of the **.c** file for all tasks (copy and paste the code not the screenshot of the code).
- Pasted **text output** or **screen shots** of the working program resulting from the testing of the program.

2. A ***.zip*** file (name of the zip file should be your student number, eg: 1012546_assignment_1.zip) containing:

a) The actual programs (.**c** source codes) with comments. Programs must be named studentid_A_Qn1.c, studentid_A_Qn2.c and so on.


Assignment 1

Due 9 Apr by 23:59 Points 100 Submitting a file upload Available until 14 Apr at 23:59

No Content

File upload Office 365

Upload a file, or choose a file you've already uploaded.
File: Browse... 1234568.docx
Browse... 1234568.zip
+ Add another file

Comments...

I agree to the tools End-User License Agreement.
This assignment submission is my own, original work

Cancel Submit assignment

Make sure two files are selected separately for submission as shown in this figure.
Give names:
studentid_Assignment_1.docx
studentid_Assignment_1.zip

End of Assignment