SENG1050 – DATA STRUCTURES

FOCUSED ASSIGNMENT 1 - DYNAMIC MEMORY ALLOCATION

OVERVIEW

- Write a program that takes in information about flights and stores the destination and date in an array of structs to be displayed. Dynamic memory allocation is used to store the strings in the struct.

OBJECTIVES

- Use dynamic memory allocation.
- Use structs.

ACADEMIC INTEGRITY AND LATE PENALTIES

- Link to Academic Integrity Information
- Link to Late Policy

EVALUATION

- The evaluation of this assignment will be done as detailed in the Marking lecture in Week 2 of the C course.

PREPARATION

- Review structs.
- View videos for Week 1.

REQUIREMENTS

Input Requirements

- The user will input ten pairs of C-style null-terminated strings about flights. The first string will contain the name of the destination. The second string will contain the date of the flight.
    - As always, prompt the user before getting the input. It is adequate (but not necessary) to prompt once for all ten pairs, as long as you are explicit about what the user should enter.

    - The two strings must be entered on two separate lines.
    - Assume the input string will be less than 30 characters long.

Struct Requirements

- Define a struct called FlightInfo that has two fields:
    - a pointer to a char (i.e. a null-terminated string) for the destination
    - a pointer to a char (i.e. a null-terminated string) for the date
- Both pointers must be set up by using malloc() once you know how long each string is (failing to use malloc will result in a mark of 0 and no further feedback). Don’t forget to add one to the length for the null-termination.
- There is **no** array in the struct.
- Define the FlightInfo struct before any of your functions. This does not violate the banning of global variables since these are data types, not variables.

Function Creation Requirements

- In main(), declare an array variable of struct FlightInfo of size 10.
- Write a function called fillFlightInfo that takes a pointer to a struct FlightInfo (which is a pointer to an element of the array) and the two flight information strings containing the destination and date information as parameters and fills in the struct's fields. Note the data type of the first parameter. You are taking in a pointer to a struct, not an array and not a struct.
    - This function is also where you **must** allocate the two blocks of memory to contain the destination string and date string.
- Write a function called printFlightInfo that takes the array of structs as a parameter and prints all of the information contained within the array in a nicely-formatted fashion, one flight per line.
    - The destination must be displayed in the first 35 characters of the line (left-justified) and the date must be displayed in the next 35 characters of the line (left-justified).
    - You must use printf() width specifiers (or research how to specify widths using cout) to help with your formatting.
- Note that fillFlightInfo must be called once for each flight (destination/date pair) while printFlightInfo must be called only once in total. You will likely want to call them from main().

Other Requirements

- You must free **all** memory that you allocated using malloc(). This must happen at the end of main().
- Do not use calloc(). Do not use the C++ new operator. Do not use C++ string class objects (since they won't work properly with malloc()).

- Do not have any other input or output.
- In **all assignments** in this course, you must adhere to the SET Coding Standards and the SENG1000 C/C++ Programming Course Requirements. If you are unsure about the latter, please refer to the C Course Notes which are available for you in the Resources area under Content.
- In **all assignments** in this course, do not have input or output that deviates from the requirements.

GIT REQUIREMENTS

- Use GitHub Classroom for revision control for your source code file.
- It is expected that you will make regular *git commit*s with meaningful commit comments (describing the changes that you successfully made since the last commit).
    - On an assignment of this size, I would expect at least 6 distinct and meaningful commits.
        - Personally, I would have many more than that.

CHECKLIST REQUIREMENTS

- Create a requirements checklist. This should contain the specific requirements from this assignment as well as any relevant requirements that have been covered in lecture or that are found in the SET Coding Standards or SET Submission Standards. Do it in whatever form you wish. Hand in your completed checklist in PDF form as checklist.pdf. Not having this checklist will result in a cap of 80 on your mark.

FILE NAMING REQUIREMENTS

- You must call your source file f1.cpp.
- You must call your checklist checklist.pdf.

SUBMISSION REQUIREMENTS

- Do not hand in any other files.
- Submit your files to the *DS: Focused Assignment 1* Assignment Submission Folder.
- Once you have submitted your file, make sure that you've received the eConestoga e-mail confirming your submission. Do not submit that e-mail (simply keep it for your own records until you get your mark).