Tutorial 1: Getting Started

ELEC 278: Fundamentals of Information Structures

The learning goals for Tutorial 1 are:

- C variables and variable management.
- C structures, enumeration, printing, conditions, and cases.

Problem 1. In the following C program, identify the scope of the variables `x` and `ptr`, and describe how their lifetimes relate. Does this program have an object lifetime bug? How do you know?

```
void test(int *ptr) {
    *ptr += 7;
}

int main() {
    int x = 0;
    test(&x);
    printf("%d\n", x);
}
```

The scope of `x` is the body of the function `main`, and the scope of `ptr` is the body of the function `test`. When the program starts, the lifetime of `x` starts first, and then the lifetime `ptr` starts at the start of the call to `test`; the lifetime of `ptr` ends first, when the function `test` exits, followed by the lifetime of `x` at the end of the function `main`. As `ptr` does not outlive `x`, which it points to, we know there is no lifetime bug; if it had outlived `x`, there would be potential for a lifetime bug if the pointer had been accessed after `x` went out of scope

Problem 2. Complete `main()` with a simple program to print out the tax form contents in a human-readable way.

```
#include <stdio.h>

// Always '0' through '9'.
typedef char digit;

// Numeric values match T1 form.
enum marital {
    MARRIED = 1,
    COMMON_LAW = 2,
    WIDOWED = 3,
    DIVORCED = 4,
    SEPARATED = 5,
    SINGLE = 6,
};

struct numeric_date {
    digit year[4];
    digit month[2];
```

```
    digit day[2];
};

struct tax_info {
    // Social insurance number.
    digit sin[9];
    // Always a valid date.
    struct numeric_date date_of_birth;
    // Always a valid date, or all zeros if empty.
    struct numeric_date date_of_death;
    // Always a valid enum value.
    enum marital marital_status;
};

void print_date(struct numeric_date *date) {
    for (int i = 0; i < 4; ++i) {
        printf("%c", date->year[i]);
    }
    printf("-%c%c-%c%c\n", date->month[0], date->month[1], date->day[0], date->day
    [1]);
}

int main() {
    struct tax_info my_tax_info = {
        .sin = {'1', '2', '3', '4', '5', '6', '7', '8', '9'},
        .date_of_birth = {
            .year = {'1', '9', '9', '7'},
            .month = {'0', '3'},
            .day = {'2', '9'},
        },
        .date_of_death = {
            .year = {'0', '0', '0', '0'},
            .month = {'0', '0'},
            .day = {'0', '0'},
        },
        .marital_status = SINGLE,
    };

    // TODO: print out the contents of the form.
    printf("SIN: ");
    for (int i = 0; i < 9; ++i) {
        printf("%c", my_tax_info.sin[i]);
    }

    printf("\n");

    printf("DOB: ");
    print_date(&my_tax_info.date_of_birth);

    printf("DOD: ");
    if (my_tax_info.date_of_death.day[0] == '0' && my_tax_info.date_of_death.day
    [1] == '0')
        printf("BLANK\n");
    else
        print_date(&my_tax_info.date_of_death);

    printf("STATUS: ");
    switch (my_tax_info.marital_status) {
        case MARRIED:
```

```
            printf("MARRIED\n");
            break;
        case COMMON_LAW:
            printf("COMMON LAW\n");
            break;
        case WIDOWED:
            printf("WIDOWED\n");
            break;
        case DIVORCED:
            printf("DIVORCED\n");
            break;
        case SEPARATED:
            printf("SEPARATED\n");
            break;
        case SINGLE:
            printf("SINGLE\n");
            break;
    }
    return 0;
}
```