Lab 2 - Imperative Programming Constructs in C

Objectives

Install VS Code on your computer.
Use simple imperative programming constructs (conditional statements and loops) to implement C functions.

Getting Started

1. Follow the installation instructions provided to install VS Code and get familiar with it.
2. Launch Visual Studio Code (VSCode) or the C IDE of your choice.
3. Create a folder named lab2, download lab2.c from Brightspace and place the file in lab2 folder. Open lab2 folder from VSCode as explained in VSCode installation instructions. Open lab2.c in VSCode. You are now ready to start working on your lab.
4. Note, that even though is a good practice to name the file where the main function is located main.c it is not mandatory. You can compile and rum lab2.c as you did with main.c in the installation instructions. In lab2.c, locate these assignment statements at the top of the file:

```
/* Replace these lines with your information */
const char* author = "";
const char* student_number = "";
```

Replace the empty strings with strings containing your name and student number. For example, student Jane Doe, who has student number 100111222, will edit the assignment statements to look like this:

```
const char* author = "Jane Doe";
const char* student_number = "100111222";
```

Don't change the variable names!

The “Rules”

For this lab,

- Your functions can assume that all function call arguments will be valid

Exercise 1: Alarm Clock

Revisiting Exercise 1 from lab 1, write a function named `alarm_clock`. The function has two parameters. The first parameter is an integer named `day` and encodes the day of the week: 0=Sun, 1=Mon, 2=Tue, ..., 6=Sat. The second parameter is a boolean named `vacation`, and it is true when we are on vacation. The function returns

an **integer** that indicates when the alarm will ring. On weekdays (non-vacations), the alarm should ring at 7 a.m., i.e., the function should return 7. On weekends (non-vacations) and vacations (weekdays), the alarm should ring at 10 a.m i.e., the function should return 10. On vacation (weekend), the alarm should be off, i.e., the function should return -1.

Here is the function header
```
int alarm_clock(int, _Bool);
```

Exercise 2: Count and Replace

Revisiting Exercise 2 from lab 1, write a function named `count_replace` that takes a positive integer (greater than zero). The program should then count from 1 to until the integer entered by the user, replacing multiples of 3 with "Fizz", multiples of 5 with "Buzz", and multiples of both 3 and 5 with "FizzBuzz". Your function does not return any value but should print the resulting sequence to the screen.

Here is the function header
```
void count_replace(int);
1 2 Fizz 4 Buzz Fizz 7 8 Fizz Buzz 11 Fizz 13 14 FizzBuzz 16
```

Exercise 3: Check Prime Number

Revisiting Exercise 3 from lab 1, write a function named `is_prime`. It takes a positive integer as a parameter. The function checks if the positive given number (greater than zero) is a prime number without using recursion. The function returns a boolean that is true if the number is prime.

Here is the function header
_Bool `is_prime`(int);

Exercise 4: Sum of even number in a range

Write a function named `sumEven` that takes a positive integer as a parameter. The function should calculate and return the sum of all even numbers from `1` to the function's input parameter. Recursion is not allowed.

Here is the function header
```
int sumEven(int);
```

For Example:
```
int result = sumEven(10); /// should return 30
```

Wrap Up

Submit lab2.c to Brightspace (Do not change the file name!). Ensure you submit the version of the file containing your solutions and not the unmodified file you downloaded from Brightspace! Before leaving the lab, demo your work to the TA for grades. If you do not demo your work to the TAs, the grade for the lab will be zero.

Note: It is your responsibility to keep track of the TA who graded your work and to check that your grade is on Brightspace. If the grade is not on Brightspace one day after you demo the work to the TA, contact the TA who graded your lab. Requests after the extra help session for lab2 is over will not be considered.

Grading schema:

Your lab work will be graded satisfactory (1%), marginal (0.5%), or unsatisfactory (0%):

- Satisfactory means that you made reasonable progress towards completing the exercises. Note that you do not have to finish all the exercises to receive a satisfactory grade.

- Marginal means that you made some progress towards completing the exercises, but your solutions were not sufficiently complete to warrant a satisfactory grade. This grade indicates that you may be falling behind and should take steps to remedy this situation.

- Unsatisfactory means that you made little or no progress towards completing the lab exercises. This indicates that you are likely having difficulty understanding important concepts and should seek help from your instructor as soon as possible. You will also receive unsatisfactory if it is apparent to the TA that you did not do enough of the lab work on your own; that is, you relied on your colleagues to explain the exercises and provide solutions (approach, algorithms, or code).