# CPEN 432: Homework Assignment 5

Deadline: 11:59 PM, 4 April, 2022

## 1 Constant Bandwidth Server [24 points]

A control application consists of two periodic tasks with worst-case computation times $C_1 = 8ms$ and $C_2 = 6ms$, and periods $T_1 = 20ms$ and $T_2 = 30ms$. Moreover, the system includes two interrupt handling routines $ISR_1$ and $ISR_2$ with average computation times of $1.0ms$ and $1.4ms$, respectively.

We want to schedule two separate constant bandwidth servers $CBS_1$ and $CBS_2$ for serving the two interrupt handling routines, respectively.

Based on profiling, we expect the first interrupt to occur more frequently, so the goal is to ensure that $U_{CBS1} = 2 \times U_{CBS2}$. Also, based on profiling, the context switch cost is supposed to be $20\mu s$.

Given the aforementioned constraints, and using results from Section 6.9.6 in the textbook (which we also discussed in Lecture 17), compute parameters $Q_{CBS1}$, $Q_{CBS2}$, $T_{CBS1}$, and $T_{CBS}2$ such that the average response time of the interrupts is minimized. Also compute and report the resulting average response times of the interrupts.

## 2 WCET Analysis [26 points]

Consider the function `check_password` given below that takes two arguments: a user ID `uid` and candidate password `pwd` (both modeled as `int`s for simplicity). This function checks that password against a list of user IDs and passwords stored in an array, returning 1 if the password matches and 0 otherwise.

```
struct entry {
  int user;
  int pass;
};

typedef struct entry entry_t;
```

```
entry_t all_pwds[1000];

int check_password(int uid, int pwd) {
  int i = 0;
  int retval = 0;

  while(i < 1000) {
    if (all_pwds[i].user == uid && all_pwds[i].pass == pwd) {
      retval = 1;
      break;
    }
    i++;
  }

  return retval;
}
```

## 2.1 Control-Flow Graph [16 points]

Draw the control-flow graph of the function `check_password`. State the number of nodes (basic blocks) in the CFG. (Remember that each conditional statement is considered a single basic block by itself.) Also state the number of paths from entry point to exit point (ignore path feasibility).

## 2.2 Side Channels [10]

Suppose the array `all_pwds` is sorted based on passwords (either increasing or decreasing order). In this question, we explore if an external client that calls `check_password` can infer anything about the passwords stored in `all_pwds` by repeatedly calling it and recording the execution time of check password. Figuring out secret data from "physical" information, such as running time, is known as a *side-channel attack*.

In each of the following two cases, what, if anything, can the client infer about the passwords in `all_pwds`?

1. The client has exactly one (uid, password) pair present in `all_pwds`
2. The client has NO (uid, password) pairs present in `all_pwds`

Assume that the client knows the program but not the contents of the array `all_pwds`.