Homework #6

Illya Starikov

Due Date: November 3rd, 2016

Problem #1

```
#include <stdio.h>
#include <reg51.h>

unsigned char input();
void delay(int time) {} // To prevent compiler errors
int power(int base, int exponent) reentrant;

typedef enum { false, true } bool;

int main() {
    unsigned char input, i;
    unsigned char sum, asciiInput;

    P2 = 0x00; // output port
    P1 = 0xFF; // Input Port

    do {
        sum = 0;
        for (i = 3; i > 0 && sum < 255; i--) {
            asciiInput = input();
            input = asciiInput & 0x0F; // convert to HEX

            sum += input * power(10, i - 1); // during execution: 100*input ->
                                             // 10*input -> 1*input

            delay(256);
        }

        if (sum <= 255) {
```

```
            P2 = sum;
        } else {
            CY = true;
            delay(1024);
        }
    } while (true);

    return 0;
}

int power(int base, int exponent) reentrant {
    if (exponent == 0) {
        return 1;
    }

    return base * power(base, exponent - 1);
}


unsigned char input() {
    return P1;
}
```

Problem #2

```
#include <stdio.h>
#include <reg51.h>

typedef enum { false, true } bool;
sbit readFromROM = P2^1;

unsigned char fibonacci(const unsigned char number) reentrant;
unsigned char get_input();

void delay(int microsecond) {} // To prevent compiler errors

int main() {
    code unsigned char sequence[] = { 0, 1, 1, 2, 3, 5, 8, 13, 21,
                                      34, 55, 89, 144, 233, 377 };
    unsigned char input, i;

    P3 = 0x00; // Output port
```

```
    P1 = 0x0FF; // Input port

    input = get_input();

    for (i = 0; i < input; i++) {
        if (readFromROM) {
            P3 = sequence[i];
        } else {
            P3 = fibonacci(i);
        }

        delay(27250);
    }

    if (i > 14) {
        CY = true;
    }

    return 0;
}

unsigned char fibonacci(const unsigned char number) reentrant {
    if (number == 0) {
        return 0;
    } else if (number == 1) {
        return 1;
    }

    return fibonacci(number - 1) + fibonacci(number - 2);
}

unsigned char get_input() {
    unsigned char input;

    do {
        input = P1 & 0x0F; // to clear upper four bits
    } while (input == 0);

    return input;
}
```

Problem #3

```
char swap_bits(const char byte) {
    char low = byte << 4,
    high = byte >> 4;

    return low | high;
}
```

Problem #4

```
#include <stdio.h>
#include <reg51.h>

unsigned char getInput();
void output(const char low, const char high);

int main() {
    unsigned char input = getInput();
    unsigned char low, high;

    P0 = 0x0FF; // input port

    if (input <= 99) {
        high = (input & 0x0F0) >> 4; // The clears the low bits,
        // but it's still the tens place. shift bits to make make it a ones place
        low = input & 0x0F;
        output(low, high);
    } else {
        P0 = 0xFF;
        P1 = 0xFF;
    }

    return 0;
}

unsigned char getInput() {
    return P0;
}

void output(const char low, const char high) {
    unsigned char asciiLow = 0x30 | low,
```

```
    asciiHigh = 0x30 | high;

    P1 = asciiLow;
    P2 = asciiHigh;
}
```

Problem #5

```
#include <stdio.h>
#include <reg51.h>

int power(int base, int exponent) reentrant;
unsigned char input();
unsigned char function(const unsigned char x);

int main() {
    unsigned char x;

    P1 = 0x0FF; // input port
    x = input();

    if (x >= 0 && x <= 3) {
        P2 = function(x);
    } else  {
        P2 = 0x00;
    }

    return 0;
}

int power(int base, int exponent) reentrant {
    if (exponent == 0) {
        return 1;
    }

    return base * power(base, exponent - 1);
}

unsigned char input() {
    return P1;
}
```

```
unsigned char function(const unsigned char x) {
    return 2*power(x, 2) + 3*x + 1;
}
```