ICS Exercise 1

October 30, 2019

1 Binary Operations (20')

Consider the following C program

```c
int a = 0x80800101;
unsigned short ua = a;
int b = ua<<1;
short c = *(short *)&a;
unsigned long d = a;
int e = a^b;
int f = ~e && a;
int g = (f|(!0))^e;
int h = (f<<3)||g;
int i = a&h;
int j = (i+0x40)+(0x13>>2);
```

Assume the program will run on an **64-bit** little-endian machine (such as x86-64). Please fill in the blanks below. You should add '0x' prefix when the data is represented as hexadecimal.

Assume we have a 16-bit number $tmp = 0x1$, you just need to write $0x1$ rather than $0x0001$. It's wrong to write $tmp = 0x00000001$ because tmp is only a 16-bit number.

Expression	Hexadecimal Representation
ua	
b	
c	
d	
e	
f	
g	
h	
i	
j	

2 Bit Operations (20')

Use **one formula** to implement the functions below. You are only allowed to use the given operations(**while/for/if** is not allowed). The integer is 32 bits.

2.1

Given an unsigned integer, swap all odd bits with even bits. (10')
Example: swapAdj(23)=43.
Legal ops: & | << >>

```c
unsigned int swapAdj(unsigned int x)
{
    // fill in your codes...


}
```

2.2

Find the rightmost different bit of x and y. Set the corresponding bit to 1 and others to 0 of the return value. (10')
Example: diffRight(17, 34)=1. diffRight(80, 52)=4.
Legal ops: & ~ ^ + -

```c
int diffRight(int x, int y)
{
    // fill in your codes...


}
```

3 Reverse bits (10')

Write a function that reverses bits of a given 32 bits unsigned integer.

For example, given 00000010100101000001111010011100(2), your function should return 00111001011110000010100101000000(2).

```c
uint32_t reverseBits(uint32_t n)
{
    // fill in your codes...







}
```

4 Function Naming (30')

1) Below are two poorly named functions written by ICS students, please give them proper function names according to their functionalities. (15')

```c
int f1 (int x, int y)
{
    return ((x&y) + ((x^y)>>1));
}
```

```c
int f2 (int x, int y)
{
    int z = x - y;
    int k = (z >> 31) & 1;
    int m = x - k * z;
    return m;
}
```

Name of f1:

Name of f2:

2) Do the functions above provide their intended functionalities for all valid parameters? Why? Please explain with concrete examples. (15')

5 Find the difference among strings (20')

Given two strings `s` and `t` which consist of lowercase letters. String `t` is generated by randomly shuffling string `s` and then inserting one more letter at a random position. You are asked to find the inserted letter.

For example, we have `s = "daze"` and `t = "zelda"`, you need to find the letter `'l'` inserted to `t`. You can get 20 points by using only 2 local variable, including one `int` and one `char`. Otherwise, you can only get 10 points.

```c
char find_diff(char *s, char *t)
{
    // fill in your codes...





}
```