1. Finish your homework independently
2. Convert this docx to pdf: “stuID_name_csapp1.pdf”
   Example: ”2017010000_zhangsan_csapp1.pdf”
3. Submit this pdf: learn.tsinghua.edu.cn

练习题：2.64；2.73；2.81；

***2.64** 写出代码实现如下函数：

```
/* Return 1 when any even bit of x equals 1; 0 otherwise.
   Assume w=32 */
int any_even_one(unsigned x);
```

函数应该遵循位级整数编码规则，不过你可以假设数据类型 int 有 $w=32$ 位。

```cpp
#include <iostream>
int any_even_one(unsigned x);
int main(){
    std::cout<<any_even_one(0x82);
    std::cout<<any_even_one(0x51);
    return 0;
}

int any_even_one(unsigned x){
    return (x & 0x55555555) != 0;
}
```

****2.73** 写出具有如下原型的函数的代码：

```
/* Addition that saturates to TMin or TMax */
int saturating_add(int x, int y);
```

同正常的补码加法溢出的方式不同，当正溢出时，saturating_add 返回 *TMax*，负溢出时，返回 *TMin*。这种运算常常用在执行数字信号处理的程序中。

你的函数应该遵循位级整数编码规则。

```cpp
#include<iostream>
#include<climits>
int saturating_add(int x, int y);
int main(){
    std::cout<<saturating_add(999,-10000)<<std::endl;
    std::cout<<saturating_add(INT_MAX-200,55)<<std::endl;
    std::cout<<saturating_add(INT_MAX,10000)<<std::endl;
    std::cout<<saturating_add(INT_MIN,10000)<<std::endl;
    std::cout<<saturating_add(INT_MIN,-100)<<std::endl;
    std::cout<<saturating_add(INT_MAX-100,45)<<std::endl;
}
```

```
int saturating_add(int x, int y){
    int PosOverflow = (x > 0 && y > 0 && x + y <= 0);
    int NegOverflow =  (x < 0 && y < 0 && x + y > 0);
    int Overflow = PosOverflow | NegOverflow;
    int sum = PosOverflow & INT_MAX |
            NegOverflow & INT_MIN |
            ~Overflow & (x + y);

    return sum;
}
```

***2.81** 我们在一个 int 类型值为 32 位的机器上运行程序。这些值以补码形式表示，而且它们都是算术右移的。unsigned 类型的值也是 32 位的。

我们产生随机数 x 和 y，并且把它们转换成无符号数，显示如下：

```
/* Create some arbitrary values */
int x = random();
int y = random();
/* Convert to unsigned */
unsigned ux = (unsigned) x;
unsigned uy = (unsigned) y;
```

对于下列每个 C 表达式，你要指出表达式是否总是为 1。如果它总是为 1，那么请描述其中的数学原理。否则，列举一个使它为 0 的参数示例。

A. `(x>y) == (-x<-y)`
B. `((x+y)<<5) + x-y == 31*y+33*x`
C. `~x+~y == ~(x+y)`
D. `(int) (ux-uy) == -(y-x)`
E. `((x >> 1) << 1) <= x`

A：不总为1。如果$x = 0$，$y = Tmin$。由于$-TMin = TMin$，此时逻辑为0。

B：总为1。担心如果x极小，y极大，使得(x+y)<<5正溢出，而31*y+33*x不发生正溢出，则逻辑为0。x=1，y=7，验证为真，实际上该式符合补码运算规律。

C：不总为1。如x=1，y=0。

D：总为1。位级表示相同，而且最后都转换成int类型。

E：总为1。先右移一位再（算数）左移一位，可能会导致结果比原来小1或者相等。