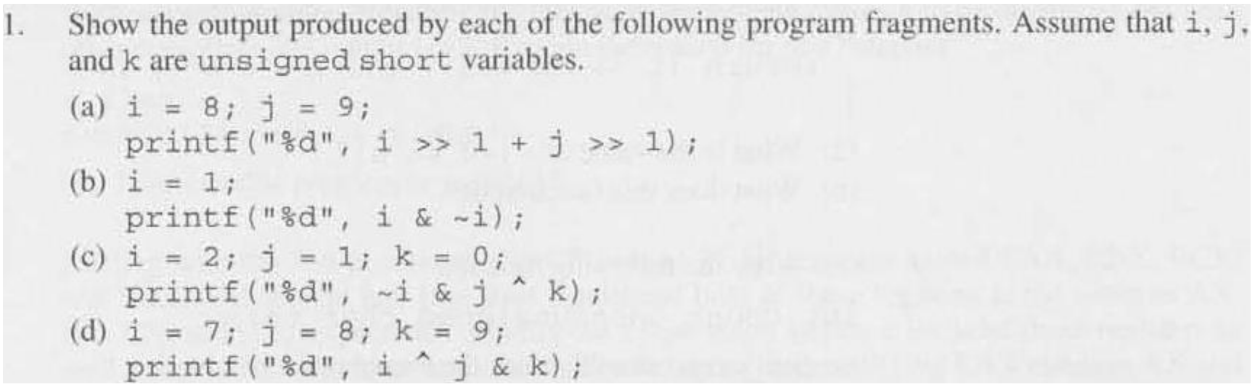
York University
Electrical Engineering and Computer Science

EECS2031: Software Tools
SU2016
Assignment #10

Chapter 20: Exercises


1. Show the output produced by each of the following program fragments. Assume that `i`, `j`, and `k` are `unsigned short` variables.

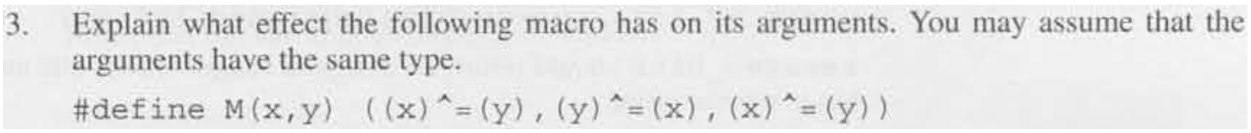
```
(a) i = 8; j = 9;
    printf("%d", i >> 1 + j >> 1);
(b) i = 1;
    printf("%d", i & ~i);
(c) i = 2; j = 1; k = 0;
    printf("%d", ~i & j ^ k);
(d) i = 7; j = 8; k = 9;
    printf("%d", i ^ j & k);
```

(a) 0

(b) 0

(c) 1

(d) 15


3. Explain what effect the following macro has on its arguments. You may assume that the arguments have the same type.

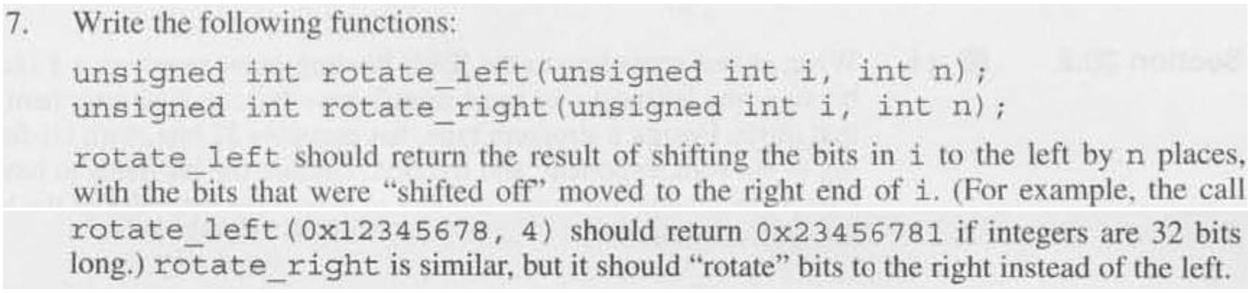
```
#define M(x,y)  ((x)^=(y),(y)^=(x),(x)^=(y))
```

The macro uses the exclusive-or operator to swap the values of its two arguments, taking advantage of the fact that (a XOR b) XOR b is equal to a. Here's how the process works:

x is assigned x XOR y

y is assigned y XOR (x XOR y), which is x

x is assigned (x XOR y) XOR x, which is y


7. Write the following functions:

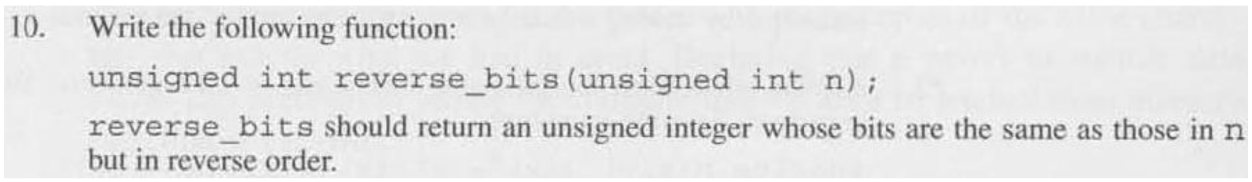
```
unsigned int rotate_left(unsigned int i, int n);
unsigned int rotate_right(unsigned int i, int n);
```

`rotate_left` should return the result of shifting the bits in `i` to the left by `n` places, with the bits that were "shifted off" moved to the right end of `i`. (For example, the call `rotate_left(0x12345678, 4)` should return `0x23456781` if integers are 32 bits long.) `rotate_right` is similar, but it should "rotate" bits to the right instead of the left.

```c
#define HIGH_BIT (~(~0U >> 1))
#define LOW_BIT  1

unsigned int rotate_left(unsigned int i, int n)
{
  while (n-- > 0)
    i = (i << 1) | (i & HIGH_BIT ? LOW_BIT : 0);
  return i;
}

unsigned int rotate_right(unsigned int i, int n)
{
  while (n-- > 0)
    i = (i >> 1) | (i & LOW_BIT ? HIGH_BIT : 0);
  return i;
}
```



10. Write the following function:

```c
unsigned int reverse_bits(unsigned int n);
```

`reverse_bits` should return an unsigned integer whose bits are the same as those in `n` but in reverse order.

```c
unsigned int reverse_bits(unsigned int n)
{
  unsigned int i, r = 0;

  for (i = 1; i > 0; i <<= 1, n >>= 1)
    r = (r << 1) | (n & 1);
  return r;
}
```