1. **Answer: (10 points)**

```
1 unsigned replace_byte (unsigned x, int i, unsigned char b)
2 {
3     int itimes8 = i << 3;
4     unsigned mask = 0xFF << itimes8;
5     return (x & ~mask) | (b << itimes8);
6 }
```

2. **Answer: (4*3=12 points)**

A. `!!x`

B. `!!~x`

C. `!!((x & 0xFF)`

D. `!!((~x & (0xFF << ((sizeof(int)-1)<<3)))`

3. **Answer: (10 points)**

```
/* Return 1 when any odd bit of x equals 1; 0 otherwise. Assume w=32 */
int any_odd_one(unsigned x) {
    /* Use mask to select odd bits */
    return (x&0xAAAAAAAA) != 0;
}
```
4. Answer: (5+8=13 points)
   A. The function does not perform any sign extension. For example, if we attempt to extract
   byte 0 from word 0xFF, we will get 255, rather than -1.
   B. The following code uses a well-known trick for using shifts to isolate a particular range of
   bits and to perform sign extension at the same time. First, we perform a left shift so that
   the most significant bit of the desired byte is at bit position 31. Then we right shift by 24,
   moving the byte into the proper position and performing sign extension at the same time.

   ```c
   1 int xbyte(packed_t word, int bytenum)
   2 {
   3   int left = word << ((3-bytenum) << 3);
   4   return left >> 24;
   5 }
   ```

5. Answer: (10 points)

   ```c
   1 /* Determine whether arguments can be subtracted without overflow */
   2 int tsub_ok(int x, int y) {
   3   int z = x-y;
   4   int sx = x>>31; // sign of x
   5   int sy = y>>31; // sign of y
   6   int sz = z>>31; // sign of result x-y
   7   return !((sx&~sy&~sz)|(~sx&sy&sz));
   8 }
   ```

6. Answer: (4*3=12 points)
   (a) 21; (b) 1C; (c) 110110.001; (d) 1A.C

7. Answer: (3*4=12 points)
   (a) 11011 (No overflow); (b) 01110 (Overflow); (c) 01010 (No overflow)

8. Answer: (3*3=9 points)

   c: 0010 0011
   d: 0100 0111 // ascii code of 'G'
   x: 0x FF FF FF D6 // two's complement representation

2
9. Answer: (12 points)

#include <stdio.h>

main()
{
    char n[4];
    int x;
    int i;

    printf("Enter a 3-digit non-negative number: ");
    scanf("%s", n);

    x = 0;
    for (i=0; i<3; i++) {
        x = 10*x + (n[i]-48);
    }

    printf("The number is %i \n", x);
}