1. Write an extended asm statement to reverse the order of the bytes within a 32-bit C variable named "x32".

```
asm   (
      "REV  %[reg1],%[reg2]"
      : [reg1] "=r" (x32)         // Output Operand
      : [reg2] "r"  (x32)         // Input Operand
      ) ;
```

2. Write an extended asm statement to rotate the contents of a 64-bit C variable named "x64" right by 1 bit position.

```
asm   (
      "LSRS  %[mshalf],%[mshalf],1                \n\t"
      "ORR   %[mshalf],%[mshalf],%[lshalf],LSL 31 \n\t"
      "RRX   %[lshalf],%[lshalf]                  "
:     [mshalf] "+r" (((uint32_t *) &x64)[1]),
      [lshalf] "+r" (((uint32_t *) &x64)[0])
:     // No InputOperands
:     "cc"  // Clobbers the C flag
) ;
```

3. Write an extended asm statement that stores into variable "count" the number of leading zeroes in the 32-bit C variable named "x32".

```
asm   (
      “CLZ  %[reg1],%[reg2]"
      : [reg1] "=r" (count)       // Output Operand
      : [reg2] "r"  (x32)         // Input Operand
      ) ;
```

4. Complete the C inline functions shown below for each of the instructions BFC, BFI, SBFX and UBFX. Use an extended asm statement that specifies the bit-field position and width using integer constants.

(a) static inline uint32_t BFC(uint32_t src, uint32_t lsb, uint32_t width)

```
static inline uint32_t BFC(uint32_t word, int lsb, int len)
      {
         asm     (
            "BFC %[reg1],%[const1],%[const2]"
            : [reg1] "+r" (word)  // Output Operand
            : [const1] "i" (lsb), // Input Operand
              [const2] "i" (len)  // Input Operand
            ) ;

         return word ;
      }
```

(b) static inline uint32_t UBFX(uint32_t src, uint32_t lsb, uint32_t width)

```
static inline uint32_t UBFX(uint32_t word, int lsb, int len)
      {
      uint32_t result ;

         asm     (
            "UBFX %[reg1],%[reg2],%[const1],%[const2]"
            : [reg1] "=r" (result)     // Output Operand
            : [reg2] "r" (word),  // Input Operand
              [const1] "i" (lsb), // Input Operand
              [const2] "i" (len)  // Input Operand
            ) ;

         return result ;
      }
```

(c) static inline int32_t SBFX(uint32_t src, uint32_t lsb, uint32_t width)

```
static inline int32_t SBFX(uint32_t word, int lsb, int len)
      {
      int32_t result ;

         asm     (
            "SBFX %[reg1],%[reg2],%[const1],%[const2]"
            : [reg1] "=r" (result)     // Output Operand
            : [reg2] "r" (word),  // Input Operand
              [const1] "i" (lsb), // Input Operand
              [const2] "i" (len)  // Input Operand
            ) ;

      return result ;
      }
```

(d) static inline uint32_t BFI(uint32_t dst, uint32_t src, uint32_t lsb, uint32_t width)

```
static inline uint32_t BFI(uint32_t dst, uint32_t src, int lsb, int len)
   {
   asm   (
         "BFI %[reg1],%[reg2],%[const1],%[const2]"
         : [reg1] "+r" (dst)      // Output Operand
         : [reg2] "r" (src),      // Input Operand
           [const1] "i" (lsb),    // Input Operand
           [const2] "i" (len)     // Input Operand
         ) ;

   return dst ;
   }
```