COMP 2401 B

Test #1 (version 2)

1. [4 marks]

   a. [3 marks]
   Answer: `0b11000110 + 37 = 2^7 + 2^6 + 2^2 + 2^1 + 37 = 235`
   `235 = 2^7 + 2^6 + 2^5 + 2^3 + 2^1 + 2^0 =` **`1110 1011`**
   Marking:
   -- 1 mark for correct approach
   -- 1 mark for correct addition result, given correct approach
   -- 1 mark for correct answer in binary, given correct approach

   b. [1 mark]
   Answer: **`235`**
   Marking:
   -- 1 mark for correct answer

2. [8 marks]

   a. [2 marks]
   Answer: `11 * 16^1 + 8 * 16^0 + 36 = 220`
   `220 = 2^7 + 2^6 + 2^4 + 2^3 + 2^2 =` **`1101 1100`**
   Marking:
   -- 1 mark for correct approach
   -- 1 mark for correct answer, given correct approach

   b. [4 marks]
   Answer: Because it's a signed char and the binary value begins with a 1, the value will be interpreted as a negative number; to get the decimal value, apply two's complement to the binary value:

	`1101 1100`
invert:	`0010 0011`
add 1:	`0010 0100`

   convert to decimal: `2^5 + 2^2 = 36`
   negative value that is printed: **`-36`**

   Marking:
   -- 2 marks for correctly applying two's complement
   -- 2 marks for correct negative decimal value (alt: 1 mark for positive decimal value)

c. [2 marks]
Answer: `2^7 + 2^6 + 2^4 + 2^3 + 2^2 =` **220**
Marking:
-- 1 mark for correct approach
-- 1 mark for correct answer, given correct approach

3. [8 marks]
Answer: **0100 0010 1000 0111 0100 0000 0000 0000**
-- sign bit: 0
-- fixed point: `67.625 = 2^6 + 2^1 + 2^0 + 2^-1 + 2^-3`
`= 1000011.101 = 1.000011101 * 2^6`
-- exponent: `6 + 127 = 133 = 1000 0101`
-- fraction: `000011101`

Marking:
-- 1 mark for correct sign bit
-- 2 marks for correct fixed point representation
-- 2 marks for correct exponent in binary
-- 2 marks for correct fraction
-- 1 mark for correct final answer, padded with zeros to make 32 bits

4. [30 marks]

a. [6 marks]

```
typedef struct {                    // 1 mark for typedef struct
  int num;                          // 1 mark for num
  char acctType[MAX_STR_SIZE];      // 1 mark for acctType
  int  yearOpened;                  // 1 mark for yearOpened
  float balance;                    // 1 mark for balance
} BankAcctType;                     // 1 mark BankAcctType
```

b. [8 marks]
i. [2 marks] Answer: input
ii. [2 marks] Answer: output
iii. [2 marks] Answer: input-output
iv. [2 marks] Answer: input

c. [16 marks]

```
// 1 mark for looping over correct collection
  for (int i=0; i<allAccts->size; ++i) {

// 8 marks for correct condition
// -- 4 marks for correctly comparing account types
// -- 2 marks for correctly comparing years
// -- 2 marks for correctly comparing balances
    if ( strcmp(allAccts->accounts[i].acctType, acctType) == 0
         && allAccts->accounts[i].yearOpened >= year
         && allAccts->accounts[i].balance >= balance ) {

// OPTION #1:
// 5 marks for correctly adding to specAccts
// -- 2 marks for assigning from allAccts current element
// -- 2 marks for assigning to specAccts
// -- 1 mark for assigning to end of specAccts
      specAccts->accounts[specAccts->size] = allAccts->accounts[i];
// 2 marks for incrementing specAccts size
      specAccts->size++;

// OPTION #2:
// 3 marks for calling addBankAcct
// 2 marks for using specAccts as param
// 2 marks for using allAccts current element
      addBankAcct(specAccts, &(allAccts->accounts[i]));
    } //end if
  } //end for
```