COMPSCI 105 SS 04 Term Test Suggested Solution

1. (i) $231_{10} = 11100111_2 = 347_8 = E7_{16}$

231 / 2 = 115 + 1
115 / 2 = 57 + 1
57 / 2 = 28 + 1
28 / 2 = 14 + 0
14 / 2 = 7 + 0
7 / 2 = 3 + 1
3 / 2 = 1 + 1
1 / 2 = 0 + 1
Collect digits from bottom to top = $\mathbf{11100111_2}$.

$11100111_2 = 011\ 100\ 111_2 = 347_8$

$11100111_2 = 1110\ 0111_2 = E7_{16}$

(ii) $312_4 = \mathbf{54_{10}} = \mathbf{204_5}$
Convert to decimal number first:
4 * 0 + 3 = 3
4 * 3 + 1 = 12 + 1 = 13
4 * 13 + 2 = 52 + 2 = $\mathbf{54_{10}}$
Convert to base 5:
54 / 5 = 10 + 4
10 / 5 = 2 + 0
2 / 5 = 0 + 2
Collect digits from bottom to top = $\mathbf{204_5}$

2. (i) $E9FA3_{16} = 1110\ 1001\ 1111\ 1010\ 0011_2$
$= 011\ 101\ 001\ 111\ 110\ 100\ 011_2 = \mathbf{3517643_8}$

(ii) $23103_4 = 10\ 11\ 01\ 00\ 11_2 = 001\ 011\ 010\ 011_2 = \mathbf{1323_8}$

3. 10100010_2 - 11010111_2

Subtrahend:	11010111_2
1's comp:	00101000_2
Add 1:	00000001_2
2's comp:	00101001_2
Minuend:	$\underline{10100010_2}+$
Carry:	00100000
Result:	$\mathbf{11001011_2}$

4. E3 7A 00 19 05 DF 0E AB
$E3\ 7A_{16} = 1110\ 0011\ 0111\ 1010_2$
Less than 5 leading zeros => 3 bytes.
1110 1110 **10**00 1101 **10**11 $1010_2 = EE\ 8D\ BA_{16}$
~~E3 7A~~ 00 19 05 DF 0E AB
$00\ 19_{16} = 0000\ 0000\ 0001\ 1001_2$
More than 8 leading zeros => 1 byte.
$0001\ 1001_2 = 19_{16}$
~~E3 7A 00 19~~ 05 DF 0E AB

$05\ DF_{16} = 0000\ 0101\ 1101\ 1111_2$
5 leading zeros (between 5 and 8 leading zeros) => 2 bytes.
1101 0111 **10**01 $1111_2 = D7\ 9F_{16}$

~~E3 7A 00 19~~ ~~05 DF~~ 0E AB
$0E\ AB_{16} = 0000\ 1110\ 1010\ 1011_2$
Less than 5 leading zeros => 3 bytes.
1110 0000 **10**11 1010 **10**10 $1011_2 = E0\ BA\ AB_{16}$

So UTF-8 = EE 8D BA 19 D7 9F E0 BA AB_{16}

5. 1. Convert to decimal:
$9_{10} = 1001_2$
9 / 2 = 4 + 1
4 / 2 = 2 + 0
2 / 2 = 1 + 0
1 / 2 = 0 + 1
$0.125_{10} = 0.001_2$
0.125 * 2 = 0.250 => 0
0.250 * 2 = 0.500 => 0
0.500 * 2 = 1.000 => 1
Collect digits from top to bottom = 0.001_2
$9.125 = 1001.001_2$

2. Normalise the number, write it in the form $1.f * 2^{\text{true exp}}$
$1001.001_2 = 1.001001_2 * 2^3$ (shift decimal point 3 places to the left)

3. Find the sign, exponent and fraction bits:
Sign = positive number = 0.
Exponent, biased by 127 = 3 + 127 = $130_{10} = 10000010_2$
130 / 2 = 65 + 0
65 / 2 = 32 + 1
32 / 2 = 16 + 0
16 / 2 = 8 + 0
8 / 2 = 4 + 0
4 / 2 = 2 + 0
2 / 2 = 1 + 0
1 / 2 = 0 + 1
Collect the digits from bottom to top: 10000010_2
Fraction: 001 0010 0000 0000 0000 0000

4. Hence the IEEE representation is:
sxxx xxxx xfff ffff ffff ffff ffff ffff
$0100\ 0001\ 0001\ 0010\ 0000\ 0000\ 0000\ 0000_2 = \mathbf{41120000_{16}}$

6. Orders of growth from slowest to fastest:
$O(1)\ O(\log_2 n)\ O(n)\ O(n\log_2 n)\ O(n^2)\ O(n^3)\ O(2^n)$

7. (a) ADT is a collection of data and a set of operations on that data.

(b)
```
try{
        String s;
        while(s=bReader.readLine()!=null){
                arList.add(s);
        }
}
catch(Exception e){}
```

8. $x = 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_2$
$y = 16 + x = 15 = 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1111_2$

x: $1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_2$
y: $0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1111_2$
x&y: $0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1111_2 = 15_{10}$

x: $1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_2$
y: $0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1111_2$
x|y: $1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_2 = -1_{10}$

x: $1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111_2$
y: $0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 0000\ 1111_2$
x^y: $1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 0000_2 = -16_{10}$

x << 15 = ~~11111111111111~~ $1111\ 1111\ 1111\ 1111\ 1000\ 0000\ 0000\ 0000_2$
$= -2^{15} = -32{,}678$

x >> 15 = $1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111\ 1111$ ~~11111111111111~~$_2$
$= -1$

Output is therefore:
x & y = 15
x | y = -1
x ^ y = -16
x << y = -32,678
x >> y = -1

9.
```
public int myPower( int a, int k ){
        if ( k == 1 )
                return a;
        else {
                int t = myPower( a, k/2 );
                if ( ( k%2 ) == 0 )
                        return t * t;
                else
                        return t * t * a;
        }
}
```

10.
```
if ( index == 1){
        head = head.getNext();
```

```
            return;
}
Node prev = null;
Node curr = head;
while( index > 1 ){
            prev = curr;
            curr = curr.getNext();
            index = index – 1;
}
prev.setNext( curr.getNext() );
numItems--;
```