1a.) $3x^3 + x^2 \log n + 2x^2 + 3\log n$

* $3x^3$ = highest time complexity

$= 3x^3 + 2x^2 + x^2 \log n + 3\log n$

$= 3x^3 + 2x^2 + \log n(x^2 + 3)$

//dropping the lower terms

$= 3x^3 + \cancel{2x^2} + \cancel{\log n(x^2+3)}$

$= 3x^3$

//dropping constants

$= x^3 \rightarrow O(x^3)$ ✓

1b.) $2x^2 + (\log x)^2 + \log x + 8$

* $2x^2$ = highest time complexity

$= 2x^2 + (\log x)^2 + \log x + 8$

$= 2x^2 + \log x(\log x + 1) + 8$

//dropping lower terms

$= 2x^2 + \cancel{\log x(\log x + 1)} + \cancel{8}$

$= 2x^2$

//dropping constants

$= x^2 \rightarrow O(x^2)$ ✓

```
~~2.) procedure twoSmallestInt (int arr[], size)~~
        ~~if (size<2) //list must contain at least 2 elements~~
            ~~return~~
        ~~for (i=0 to size)~~
            ~~if (arr[i] < firstNum)~~
                ~~secondNum = firstNum~~
                ~~FirstNum = arr[i]~~
            ~~else if (arr[i] < second Num)~~
                ~~secondNum = arr[i]~~
        if
```

```
2.) procedure TwoSmallest (int arr[], int size)
        //at least two elements
        if(size < 2)
            return
         firstNum = secondNum = max
         for(i = 0 ; i < size ; i++)
            //if element is smaller than first
            if(arr[i] < firstNum)
                secondNum = firstNum
                firstNum = arr[i]
            //if element is smaller than first
            ~~if (arr[i] < first)~~
                ~~secondNum = firstNum~~
                ~~first~~
             else if(arr[i] < secondNum)
                 secondNum = arr[i]
           if (secondNum == max)
               return
           else
               ~~cout << first & second~~
               cout << firstNum & secondNum
```

```
list: 22 3 4 55
if 22 < 3 → else statement
3 = arr[i]
if 3 < 4 → 4 = arr[i] → firstNum = 3
if 4 < 55 → 4 = arr[i] → secondNum = 4
prints: 3 & 4
```

3a.)
```
for i = 1 to n-1  // O(n-1)
    for j = 2i+1 to 2n-1  // O(2n-1-i)
        comparison operation
```
$= 2(n-1-1) + 2(n-1-2), + \dots, + 2(n-1-(n-1)$

$= 2 \cdot (n \cdot (n-1) - 1)(n-1) - (n \cdot (n-1)/2))$

$= 2 \cdot (n \cdot (n-1)/2 - (n-1) = n^2 - n - 2n + 2$

$= O(n^2)$

3b.)
```
for i = 1 to n-1  // O(n-1)
    for j = 1 to m+1  // O(m+1)
        for k = 1 to t  // O(t)
            comp. operation
    for l = 1 to n  // O(n)
        multiplication op.
```
$n-1 \cdot m+1 \cdot t + n = O(n * m * t)$

4a.)
```
procedure negative count(a1, a2, ..., an : integers
with n≥1)
    k := 0
    for i := 1 to n
        if ai < 0 then k := k+1
            return k
```

4b.)
```
int power(int a, int b)
    if (b < 0)
        return 0
    else if (b == 0)
        return 1
    else
        return a * power(a, b-1)
```

5.) GCD(324, 124)

323 = 124 · 2 + 75

124 = 75 · 1 + 49

75 = 49 · 1 + 26

49 = 26 · 1 + ~~26~~ 23

26 = 23 + 3

23 = 3 · 7 + 2

3 = 2 · 1 + 1

2 = 1 · 2 + 0

GCD = 1

1 = 323 (s) + 124 (t)

s = 43

t = -112